Nama : Oktaviani
NIM : 119140014
Kelas : PAM RB

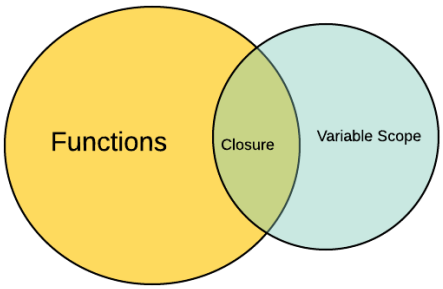
JAVASCRIPT DAN THE NEXT GENERATION JAVASCRIPT

1. CLOSURE

   Dalam JavaScript, *closure* merupakan sebuah fungsi yang dieksekusi oleh fungsi lainnya (*nested functions*) sehingga fungsi tersebut dapat mengakses atau merekam *variable* di dalam *lexical scope* pada fungsi induk (*parent function)*. *Closure* adalah kombinasi dari fungsi dan lexical environment dimana fungsi itu di deklarasikan.

   Terdapat dua poin penting dalam *closure* :

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



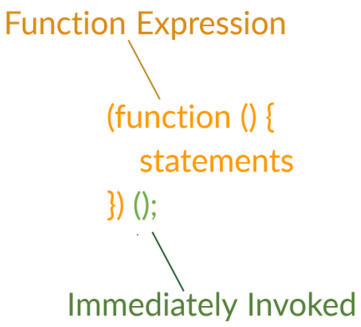
Sumber : JavaScript: Apa itu Closure?

Sumber
https://medium.com/@rendybustari/menggunakan-closure-pada-javascript-72eddfe7f02e

2. IMMEDIATELY INVOKED FUNCTION EXPRESSION

   IIFE (Immediately Invoked Function Expression) adalah fungsi JavaScript yang berjalan segera setelah didefinisikan. IIFE sangat berguna karena tidak mencemari objek global, dan merupakan cara sederhana untuk mengisolasi deklarasi variabel. Tiga konsep sederhana IIFE yaitu :

   1. Segera Dipanggil - segera dijalankan
   2. Fungsi - fungsi javascript yang khas
   3. Ekspresi- ekspresi javascript adalah bagian dari kode yang hanya mengevaluasi nilai.



Sumber : https://id.quish.tv/what-is-an-iife-javascript

- Function Expression

  Komponen dalam warna oranye adalah ekspresi fungsi yang merupakan fungsi anonim yang dibungkus dalam |_+_|.
- Immediately Invoked

  Komponen berwarna hijau membuat pemanggilan langsung dari ekspresi fungsi dengan menggunakan operator pemanggilan yaitu |_+_|. Dengan kata lain, ini pada dasarnya memanggil ekspresi fungsi seperti cara anda memanggil fungsi lainnya.

Sumber

https://www.geeksforgeeks.org/javascript-immediately-invoked-function-expressions-iife/

3. FIRST-CLASS FUNCTION

   Dalam ilmu komputer, Bahasa pemrograman dapat dikatakan mendukung first-class function apabila memberlakukan function sebagai first-class object. First-class object 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 ke dalam variable, object atau array. Apabila dikaitkan dengan javascript, intinya function pada javascript adalah **first-class object**, Yang berarti **function** tersebut dapat:

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

   Sumber : https://medium.com/@aryarifqipratama/mengenal-lebih-dalam-tentang-first-class-function-pada-javascript-9d12cb11febe

4. HIGHER-ORDER FUNCTION

   Higher-Order Function adalah fungsi yang beroperasi pada fungsi lain, baik dengan mengambilnya sebagai argumen atau dengan mengembalikannya. Dengan kata sederhana, Higher-Order Function adalah fungsi yang menerima fungsi sebagai argumen atau mengembalikan fungsi sebagai output.

   Misalnya, Array.prototype.map, Array.prototype.filter dan Array.prototype.reduce adalah beberapa fungsi Tingkat Tinggi yang dibangun ke dalam bahasa.

   Contoh :

   1. Array.prototype.map
   2. Array.prototype.filter
   3. Array.prototype.reduce

   Sumber

   Understanding Higher-Order Functions in JavaScript | by Sukhjinder Arora | Bits and Pieces

5. EXECUTION CONTEXT

   Execution Context didefinisikan sebagai konteks atau lingkungan dimana javascript di eksekusi. Execution Context merupakan pembungkus yang mengelola code yang sedang dijalankan. Execution Context terbagi menjadi 2 yaitu Global Execution Context dan Local Execution Context. Dan dalam javascript terdapat 2 fase pada saat Execution Context yaitu creation phase dan execution phase.

   Sumber

   https://muhammadfahri.com/hoisting-execution-scope/

6. EXECUTION STACK

   Execution Stack juga dikenal sebagai "Calling Stack" dalam bahasa pemrograman lain, adalah Stack dengan struktur LIFO (Last in, First out), yang digunakan untuk menyimpan semua konteks eksekusi yang dibuat selama eksekusi kode.

   Sumber

   https://blog.bitsrc.io/understanding-execution-context-and-execution-stack-in-javascript-1c9ea8642dd0

7. EVENT LOOP

   Event Loop adalah sesuatu yang menarik barang keluar dari antrian dan menempatkannya ke Function Execution Stack setiap kali Function Execution Stack kosong.

   Sumber

   https://www.geeksforgeeks.org/what-is-an-event-loop-in-javascript/

8. CALLBACKS

   Callback pada Javascript adalah sebuah fungsi yang dikirimkan sebagai parameter fungsi lainnya. Biasanya callback digunakan pada sebuah fungsi yang memiliki sifat Asynchronous, yang kemudian akan dieksekusi ketika fungsi Asynchronous tersebut selesai. Sebagai contoh yang paling sering digunakan oleh aplikasi kecil adalah setTimeout(function, time), sebuah fungsi yang menerima fungsi lain yang akan dieksekusi setelah waktu (time) yang diberikan. Untuk aplikasi besar biasanya digunakan pada proses hit API (Application Programming Interface), yang biasanya direpresentasikan dengan success atau .then().

   Sumber

   https://id.quora.com/Apa-arti-callback-dalam-Javascript

9. PROMISES DAN ASYNC/AWAIT

   PROMISES

   Promise digunakan untuk melacak apakah peristiwa asinkron telah dijalankan atau tidak dan menentukan apa yang terjadi setelah peristiwa itu terjadi. Ini adalah objek yang memiliki 3 status yaitu:

   1. Pending yaitu keadaan awal atau sebelum peristiwa terjadi.
   2. Resolved yaitu keadaans etelah operasi selesai dengan sukses.
   3. Rejected yaitu jika operasi mengalami kesalahan selama eksekusi, janji gagal.

   ASYNC/AWAIT

   Async/Await digunakan untuk bekerja dengan Promises dalam fungsi asinkron. Ini pada dasarnya adalah sugar sintaksis untuk Promises. Ini hanyalah pembungkus untuk menata ulang kode dan membuat Promises lebih mudah dibaca dan digunakan. Itu membuat kode asinkron lebih mirip kode sinkron/prosedur, yang lebih mudah dipahami.

   Menunggu hanya dapat digunakan dalam fungsi async. Ini digunakan untuk memanggil fungsi async dan menunggunya untuk diselesaikan atau ditolak. menunggu memblokir eksekusi kode dalam fungsi async di mana ia berada.

   Sumber

   Difference between promise and async await in Node.js - GeeksforGeeks

Link github :

https://github.com/oktaviani119140014/Tugas-Individu-2.git