Tugas Individu 2
JavaScript Lanjutan

Justin Halim
Pengembangan Aplikasi Mobile RB
120140185

Institut Teknologi Sumatera

Tahun Ajaran 2022/2023

SOAL

Buatlah sebuah rangkuman materi javascript lanjutan dari kata kunci di bawah dan berikan contoh sendiri:

1. Closure
2. Immediately Invoked Function Expression
3. First-class function
4. Higher-order function
5. Execution Context
6. Execution Stack
7. Event Loop
8. Callbacks
9. Promises dan Async/Await

PEMBAHASAN

1. Closure
   Variabel JavaScript bisa terbagi menjadi scope global dan lokal. Variabel global bisa menjadi variabel lokal (private) dengan menggunakan fungsi closure.
   Contoh :

```
const add = (function () {
  let counter = 0;
  return function () {counter += 1; return counter}
})();

add();
add();
add();
```

2. Immediately Invoked Function Expression (IIFE)
   IIFE adalah function JavaScript yang langsung berjalan sesaat setelah di-define.
   Contoh :

```
(function () {
  // …
})();

(() => {
  // …
})();

(async () => {
  // …
})();
```

3. First-class function
   JavaScript First-class function bisa diketahui apabila fungsi-fungsi dalam code bersifat seperti variabel. Salah satu kasusnya, seperti suatu fungsi bisa dijadikan argumen bagi fungsi lainnya.
   Contoh :

```
const Person = {
    play:(name) => {
        return `Hey ${name} is playing`;
    },
    dance:(name) => {
        return `${name} can dance`
    },
    walk:(name) => {
        return `I am sure ${name} can walk `
    },
}

console.log(Person.play("Chibueze"));
console.log(Person.dance("Chibueze"));
console.log(Person.walk("Chibueze"));
```

4. Higher-order function
   Higher-order functions adalah fungsi yang mengembalikan sebuah fungsi atau mengambil fungsi sebagai argument.
   Contoh :

```
function doubleElements(x){
  return 2*x;
}

let DoubleResult=[1,2,3,4,5,6].map(doubleElements)

function filterElemLesThanTwo(x){
  return x<2
}

let ResultLessThanTwo=[1,2,3,4,5,6].filter(filterElemLesThanTwo)

 console.log(DoubleResult)
 console.log(ResultLessThanTwo)
```

5. Execution Context
   When a fragment of JavaScript code runs, it runs inside an execution context. There are three types of code that create a new execution context:
   - The global context is the execution context created to run the main body of your code; that is, any code that exists outside of a JavaScript function.
   - Each function is run within its own execution context. This is frequently referred to as a "local context."
   - Using the ill-advised eval() function also creates a new execution context.

Contoh :

```
let outputElem = document.getElementById("output");

let userLanguages = {
  "Mike": "en",
  "Teresa": "es"
};

function greetUser(user) {
  function localGreeting(user) {
    let greeting;
    let language = userLanguages[user];

    switch(language) {
      case "es":
        greeting = `¡Hola, ${user}!`;
        break;
      case "en":
      default:
        greeting = `Hello, ${user}!`;
        break;
    }
    return greeting;
  }
  outputElem.innerHTML += `${localGreeting(user)}<br>\r`;
}

greetUser("Mike");
greetUser("Teresa");
greetUser("Veronica");
```

6. Execution Stack
   Execution Stack atau yang biasa dikenal sebagai Call Stack, mengecek segala Execution Context yang diciptakan saat life cycle di script. Karena JavaScript adalah bahasa pemrograman yang single-threaded, JavaScript hanya bisa mengeksekusi satu task setiap waktu. Jadi, setiap fungsi dan events yang terjadi Execution Context membuat setiap kegiatannya.
   Contoh :

```
var name = "Victor";

function first() {
  var a = "Hi!";
  second();
  console.log(`${a} ${name}`);
}

function second() {
  var b = "Hey!";
  third();
  console.log(`${b} ${name}`);
}

function third() {
  var c = "Hello!";
  console.log(`${c} ${name}`);
}

first();
```

7. Event Loop
   Event Loop function mengecek call stack, dan apabila call stack tidak memiliki fungsi, maka callback function akan mengambil fungsi callback tertua dalam queue untuk mengeksekusi fungsi callback yang dipanggil function event loop.
   Contoh :

```
console.log('hi');

setTimeout(function() {
     console.log('freecodeCamp')
},5000);

console.log('JS')
```

8. Callbacks
   Callbacks function adalah fungsi yang dimasukkan kedalam fungsi lain, dan dipanggil untuk melakukan task didalam fungsi yang dimasuki.
   Contoh :

```
console.log('fired first');
console.log('fired second');

setTimeout(()=>{
    console.log('fired third');
},2000);

console.log('fired last');
```

9. Promises dan Async/Await
   Promises function dalam JavaScript bisa membantu menangani masalah yang dialami di Callback function. Function promises ini terdiri dari 2 function yang menjadi parameternya, yaitu resolve dan reject. Resolve terjadi saat promises diwujudkan dan reject akan menolak promises dan terjadi error.
   Async/Await function membantu dalam mempermudah memahami dan mempersingkat function Promises dan Callbacks.
   Contoh :

```
const asyncFunc = async () => {
    const response = await fetch(resource);
    const data = await response.json();
}
```

LAMPIRAN

https://www.w3schools.com/default.asp

https://developer.mozilla.org/en-US/

https://www.educative.io/answers/what-are-first-class-vs-higher-order-functions-in-javascript

https://www.freecodecamp.org/

https://github.com/120140185J/RA_PAM_TUGAS2_120140185.git