NAMA: SALDIN

NIM : 22650177

TUGAS MOBILE PROGRAMMING

LAPORAN JAVASCRIPT

1. Variabel

Variabel merupakan tempat menyimpan data. Berikut 3 cara untuk membuat variable dalam JS, yakni:

a. Menggunakan var

Kata kunci ini bersifat Global Scope, dikarenakan variabel ini dapat di akses secara global dan diberikan nilai berulang kali. Perhatikan kode berikut:

```
var age = 20
{
    var age = 40
}
var age = 50

console.log(age) // Output: 50
```

Gambar.1 variabel Var

Pada kode tersebut terdapat tiga variabel dengan nama yang sama yaitu *age,* dan diberikan nilai awal berupa 20 lalu di update menjadi 40 hingga terakhir menjadi 50. Baris kode tersebut merupakan kode yang valid dan tidak terdapat error. Namun ada kalanya kita membuat variabel dengan nama yang sama akan tetapi berada dalam ruang lingkup yang berbeda. Contohnya:

```
var nama = 'John'

function sayMyName() {
    var name = 'Heisenberg'
    console.log(nama)
}

console.log(nama) // John
sayMyName() // Fungsi dijalankan dan memiliki output: Heisenberg
console.log(nama) // Value dari variabel pertama berubah menjadi: Heisenberg
```

Gambar.2 variabel var nama sama dalam ruang lingkup berbeda

Pada kode di atas, kita mendefinisikan variabel name yang memiliki value ‘John’. Kemudian pada baris berikutnya terdapat fungsi sayMyName yang memiliki nama variabel yang sama namun dengan value ‘Heisenberg’.

Apabila kita console variabel pertama, akan didapatkan output berupa ‘John’. Namun ketika fungsi sayMyName dijalankan, kemudian kita console variabel name yang pertama, maka nilai yang semula ‘John’ berubah menjadi ‘Heisenberg’.

Hal ini tentu tidak kita inginkan, karena dapat mengakibatkan sistem yang dibangun menjadi tidak konsisten karena nilai variabel yang berubahubah. Untuk mengatasi hal tersebut Javascript menyediakan dua kata kunci lainnya yaitu let dan const.

b. Menggunakan Let

Ketika menggunakan kata kunci let, maka kita mendapatkan variabel yang bersifat Local Scope dan dapat diubah nilainya namun tidak dapat dideklarasikan ulang. Perhatikan kode berikut:

```
let age = 30
age = 50 // Melakukan perubahan nilai

{
    age = 30 // Tetap bernilai 30
}

console.log(age) // Output: 50
```

Gambar.3 Variabel Let

Kode tersebut berjalan tanpa adanya error, namun berbeda jika kita mendeklarasikan ulang variabel tersebut:

```
let age = 30
let age = 40
console.log(age) // SyntaxError:Identifier 'age' has alre
```

Gambar.4 Variabel Let yang salah

Terjadi error dikarenakan kata kunci let tidak mengizinkan kita untuk mendeklarasikan ulang variabel dengan nama yang sama.

c. Menggunakan const

Sama seperti let, kata kunci const menghasilkan variabel yang memiliki sifat Local Scope namun tidak dapat diubah nilainya dan tidak bisa dideklarasikan ulang. Hal ini sesuai dengan namanya yaitu Constanta. Ketika menggunakan kata kunci const, variabel yang dihasilkan akan menjadi lebih strict dan membuatnya konsisten. Berikut ini contoh penerapannya:

```
// 1. Kita mendeklarasikan ulang variabel dan melakukan update terhadap valuenya
const data = 'John'
const data = 'Doe'
console.log(data) // SyntaxError: Identifier 'data' has already been declared

// 2. Kita melakukan update data terhadap variabel const
const data = 'John'
data = 'Doe'
console.log(data) // TypeError: Assignment to constant variable.
```

Gambar.5 kesalahan dalam menggunakan const

Berdasarkan point di atas, apabila ingin membuat variabel direkomendasikan menggunakan kata kunci let ataupun const. Hal ini bertujuan untuk meminimalisir terjadinya duplikasi dan perubahan nilai pada variabel yang tidak kita inginkan.

2. Array

Array merupakan struktur data pada Javascript yang bertujuan untuk menyimpan beberapa nilai dan tipe data dalam satu variabel. Contohnya:

```
const fruits = ['apel', 'pisang', 'jeruk']
```

Pada kode di atas kita membuat variabel yang memiliki tipe data array dan menampung value berupa apel, pisang, dan jeruk. Untuk mengakses value yang berada di dalam array, kita bisa memanfaatkan indeks array. Indeks atau Index dalam array akan selalu dimulai dari 0, maka apabila ingin mengakses value dalam array tersebut, kita bisa melakukannya seperti ini:

```
const fruits = ['apel', 'pisang', 'jeruk']

console.log(fruits[0]) // Output: apel
console.log(fruits[1]) // Output: pisang
console.log(fruits[2]) // Output: jeruk
```

Array juga dapat menyimpan beberapa tipe data kompleks sekaligus seperti yang diterangkan di awal.

```
const someArray = ['String', true, 3, {
    name: "John",
    age: 30,
    address: "123 Main St",
    sayHello: function () {
        console.log("Hello, my name is " + this.name);
    },
    isLogin: true
}, [1, 2, 3, 4, 5], function () {
    console.log("Hello World");
}]
```

3. Method Array

Javascript menyediakan fungsi khusus yang berkaitan dengan array, salah satunya ialah fungsi map.

```
fruits.map((result) => {
    console.log(result)
})
```

Pertama kita panggil nama array yang kita miliki, yaitu array bernama fruits. Kemudian kita melakukan proses chaining method dengan memanggil fungsi map. Fungsi map bertujuan untuk membuat array baru dan hasil dari fungsi tersebut akan ditampung kedalam variabel yang kita beri nama result. Kemudian pada langkah terakhir kita cetak nilai dari variabel result tersebut. Maka kita telah berhasil mencetak nilai secara keseluruhan dari dalam array tanpa harus melakukannya satu persatu dengan indeks. Selain menggunakan kata kunci map, Javascript menyediakan beberapa fungsi lainnya, seperti:

a. forEach()
b. length()
c. pop()
d. push()
e. shift()

4. Perulangan

Perulangan pada Javascript merupakan salah satu basic yang wajib kamu kuasai,kenapa demikian? Perhatikan kode berikut:

```
console.log(1)
console.log(2)
console.log(3)
console.log(4)
console.log(5)
```

Terlihat kita ingin mencetak nilai console secara berulang sebanyak 5 kali. Namun bagaimana jika ingin mencetak nilai console tersebut hingga sampai 100 atau bahkan 1000 percobaan? Tentu apabila menuliskannya satu persatu maka akan menghabiskan waktu dan sangat tidak efisien. Untuk mengatasi hal tersebut, kita akan memanfaatkan fitur perulangan pada Javascript yaitu menggunakan kata kunci for.

```
for (let index = 0; index <= 100; index++) {
    console.log(index)
}
```

Pada fungsi for tersebut menerima tiga parameter yakni:

a. Parameter pertama: Merupakan inisialisasi variabel dan parameter ini akan selalu berjalan satu kali sebelum perulangan di mulai. Pada kode di atas kita mendefinisikan sebuah variabel dengan nama index dan bernilai 0.
b. Parameter kedua: Merupakan kondisi mengapa perulangan tersebut terjadi. Pada parameter kedua, kita definisikan kondisi yaitu variabel pada parameter pertama tidak boleh lebih atau sama dengan 100.
c. Parameter ketiga: Merupakan sebuah kondisi yang di mana akan selalu dieksekusi setiap kali perulangan tersebut dijalankan. Pada parameter ketiga, kita melakukan modifikasi variabel index dengan melakukan proses increment variabel dengan operator ++, yang dimana berfungsi untuk menambahkan satu nilai setiap perulangan. Maka outputnya adalah nilai dari variabel tersebut akan terus bertambah sesuai dengan kondisi yang kita atur pada parameter kedua.

Selain kata kunci for, Javascript menyediakan juga beberapa kata kunci lainnya yaitu: While, Do While, For In, dan For Of

5. Percabangan

Percabangan atau pengecekan kondisi adalah cara agar program yang kita bangun berjalan sesuai dengan kondisi yang diinginkan. Javascript menyediakan beberapa kata kunci untuk melakukan percabangan yang salah satunya adalah if else statement.

```
for (let index = 0; index <= 100; index++) {
    if (index % 2 == 0) {
        console.log(`Angka ${index} adalah Genap`)
    } else {
        console.log(`Angka ${index} adalah Ganjil`)
    }
}
```

Dari kode di atas kita mengkombinasikan tiga point yaitu:

Melakukan perulangan sebanyak 100 percobaan. Lalu melakukan percabangan dimana, jika hasil sisa bagi variabel index dengan 2 adalah 0, maka kita cetak isi dari variabel tersebut dengan pesan angka tersebut genap. Namun jika hasil pada kondisi pertama tidak memenuhi syarat, bisa dipastikan angka tersebut merupakan bilangan ganjil.

Berikut ini beberapa kata kunci yang sering digunakan untuk melakukan percabangan atau pengecekan kondisi: if else, switch case, ternary operator.

6. Function

Function pada Javascript adalah kumpulan dari baris kode yang berfungsi untuk menyelesaikan task tertentu. Pada point-point sebelumnya kita sudah menggunakan fungsi seperti melakukan perulangan kata kunci for atau melakukan percabangan dengan kata kunci if. Namun bagaimana jika kita ingin membuat fungsi kita sendiri? Javascript menjawab pertanyaan tersebut dengan cara:

a. Menggunakan function

```
function sum(a, b) {
    return a + b
}
console.log(sum(1, 2)) // Output: 3
```

b. Menggunakan arrow function

```
const sum = (a, b) => {
    return a + b;
}
console.log(sum(1, 2)); // Output: 3
```

Pada kode di atas kita membuat variabel yang menampung data dari hasil fungsi sum, Karena ketika membuat fungsi dengan kata kunci function, Javascript memberikan sifat Global Scope yang dimana kita bisa mengakses fungsi tersebut sebelum dideklarasikan. Berbeda ketika kita menggunakan

arrow function, maka akan terjadi error karena arrow function tidak memiliki sifat Global Scope.

7. Operator

Operator	Nama	Contoh	Hasil
-	Subtraction	x= 5	x = 5
+	Addition	x = 5 + 1	x = 6
*	Multiplication	x = 6 * 2	x = 12
++	Increment	x++	x = 13
--	Decrement	x—	x= 12
%	Modulus (Sisa bagi)	x = 10 % 2 == 0	true

- JavaScript Arithmetic Operators

Operator	Nama	Contoh	Hasil
==	Equal to	x == 8	false
===	Equal value and type	x === "5"	false
!=	Not equal	x != 8	true
!==	Not equal value or type	x !== "5"	true
>	Greater than	x > 8	false
<	Less than	x < 8	true
>=	Greater or equal to	x >= 8	false
<=	Less or equal to	x <= 8	true

- JavaScript Arithmetic Operators

Operator	Nama	Contoh
&&	AND	(x < 10 && y > 1) is true
||	OR	(x === 5 || y === 5) is false
!	NOT	!(x === y) is true