Pertemuan 21 - Recap Materi Javascript

Pendahuluan

Javascript merupakan bahasa pemrograman yang High Level dan memiliki integrasi dengan web tetapi juga dapat digunakan untuk bagian server juga, bahasa ini merupakan `interpreted`, cara pendefinisian Javascript di web ada dua:

1. Dengan `<script>`, ini biasanya ditaruh di bagian bawah `<body>`

   ```
   <script>
       <!-- Kode Javascript -->
   </script>
   ```

2. Dengan file Javascript sendiri (`.js`) untuk menambahkan file tersebut ke HTML bisa menggunakan elemen `<script>` tetapi ditambah attribute `src` dan `type`

   ```
   <script type="text/javascript" src="nama-file.js"></script>
   ```

Sebelum kita memperdalam pemahaman Javascript kita akan mengingat sedikit materi dari sebelumnya.

Tipe data secara Umum dan Tipe data yang hanya ada di Javascript

Sebelum kita memulai kita mengingat dulu tipe data, secara umum tipe data digunakan untuk klasifikasi yang menentukan tipe nilai yang dimiliki variabel dan tipe operasi matematis, relasional, atau logika apa yang dapat diterapkan padanya tanpa menyebabkan kesalahan.

Tipe data ini secara simpel suatu cara untuk komputer untuk mengetahui suatu tipe di suatu data agar dapat melakukan operasi dengan benar.

Tipe data umum

- `Char/Character` :
  Merupakan tipe data yang merepresentasikan satu suatu 1 huruf seperti `"a"`.
- `String` :
  Merupakan tipe data yang merepresentasikan kumpulan `Char` atau huruf yang lebih dari 1, contoh `"Hello, World!"`.
- `Boolean` :
  Merupakan tipe data yang merepresentasikan logika seperti `True` yang artinya benar dan `False` yang artinya salah.
- `Integer` :
  Merupakan tipe data yang merepresentasikan bilangan bulat atau angka yang tidak memiliki koma didalamnya.

- `Float` :
  Merupakan tipe data yang merepresentasikan angka yang memiliki koma didalamnya.

Tipe data yang ada di Javascript

Tipe data di Javascript ada 2 yaitu `Primitive` / Tipe data dasar, dan `Objects`. Tipe data Primitif sama dengan yang tipe data umum tetapi sedikit berbeda pada `Integer` dan `Float` karena di Javascript tipe data angka hanya ada satu `Number`, tipe data ini hanya dapat menyimpan satu data sedangkan `Object` dapat menyimpan banyak.

Primitive Type

- `Char` :
  Sama seperti tipe data yang umum.
- `Boolean` :
  Sama seperti tipe data yang umum.
- `Number` :
  Merupakan tipe data yang merepresentasikan suatu angka.
- `undefined` :
  Merupakan tipe data yang kosong atau belum di inisialisasikan.
- `null` :
  Merupakan tipe data yang kosong.
- `function` :
  Merupakan tipe data yang merepresentasikan suatu function, sering disebut `Anonymous function` atau mungkin Lamda (λ) expression tergantung konteksnya.

Object Type

- `Object` :
  Merupakan tipe data dasar di Javascript, sering digunakan untuk merepresentasikan `Class Object`, `String Object`, `Array Object`, dll.

- `String` :
  Sama seperti dengan tipe data yang umum tetapi di Javascript ini sering di representasikan sebagai `String Object`.

  ```
  let jsString = "Hello, World!";
  ```

- `Array` :
  Merupakan tipe data yang digunakan untuk menyimpan banyak data yang secara kontinu di memori.

  ```
  const jsArray = ['Zagar', 'Dangure', 1, null, undefined, true, fase];
  ```

- `Javascript Object` :
  Merupakan tipe data yang menyimpan banyak data dengan konsep Key-Value Pair jadi ada kunci dan isi di setiap data yang disimpan.

```
const jsObject = {
    //Key     Value
    "pesan": "Hello, World"
};
```

Operasi Matematika dan Perbandingan di Pemograman

Pemrograman secara tidak langsung merupakan alat Matematika karena dulunya hanya didesain untuk melakukan suatu kalkulasi yang berat, semua operasi ini akan menghasilkan sesuatu jadi untuk contoh kita beri variabel juga untuk merepresentasikan data.

Operasi Matematika

- Penjumlahan `+`
  contoh : c = a + b
- Pengurangan `-`
  contoh : c = a - b
- Pengkalian `*`
  contoh : c = a * b
- Pembagian `/`
  contoh : c = a / b

Operasi perbandingan

Operasi perbandingan ini memberikan kita tipe data `Boolean` yang nanti bisa kita pakai untuk melakukan penjalanan kode yang dinamis sesuai hasil perbandingan.

- Persamaan / Equal `==`
  contoh : c = a == b
- Lebih besar / Bigger than `>`
  contoh : c = a > b
- Lebih kecil / Less than `<`
  contoh : c = a < b
- Lebih besar sama dengan / Bigger equal `>=`
  contoh : c = a >= b
- Lebih kecil sama dengan / Less equal `<=`
  contoh : c = a <= b

Kode pertama

Sebelum kita mengingat dari materi sebelumnya kita mulai dengan yang paling sederhana yaitu men-print "Hello, World" ke browser, di Javascript ada dua ada yang lewat halamannya langsung atau dengan console di browser.

```
// Untuk menulis `Hello, World` ke HTML
document.write('Hello, World');
```

```
// Untuk menulis `Hello, World` ke console
console.log('Hello, World');
```

Untuk mengecek console browser bisa dilakukan dengan membuka `dev tool` kalau di firefox bisa menekan `F12` lalu muncul banyak tab di bagian bawah browser, salah satu tab tersebut ialah console.

Variable

Variable merupakan cara kita menyimpan suatu data di memori komputer, secara simpel variabel ini hanya merepresentasikan alamat memori di komputer dengan teks, di Javascript ada 3 untuk melakukan pendeklarasian variabel, masing-masing memiliki kelebihan dan kekurangan.

```
var a = 1;
let b = 1;
const c = 1;
```

- `var` :
  Variabel ini bersifat global jadi dapat diakses dimana saja walaupun di definisikan bukan di global, jadi untuk aman jauhi ini.
- `let` :
  Variabel ini hanya ada di `scope` yang di definisikan.
- `const` atau Constant (Konstanta)
  Variabel ini hanya ada di `scope` yang di definisikan tetapi variabel ini tidak dapat diubah isinya kecuali tipe data tertentu yang dapat menambahkan data kedalam variabel tersebut tanpa mengubah isi keseluruhan.

Kita juga dapat memberikan data yang berbeda kedalam variabel yang sudah di definisikan kalau variabel tersebut bukan `const`, kalau misal mengganti variabel `const` nanti akan error.

```
// Melakukan pendefinisian a di scope global
let a = 1;

// Menganti isi variabel a menjadi 3
a = 3;
```

Scope

Nah apa sih `Scope`, `Scope` itu merupakan block kode yang mendefinisikan panjang hidupnya suatu variabel di Javascript block tersebut dapat dimulai dengan `{` dan ditutup `}`, karena Javascript merupakan bahasa High Level jadi menggunakan konsep `Scope`, secara sederhana di dalam file Javascript itu merupakan `Scope` Global karena dapat diakses dimana saja

```
// Scope Global
let a = 1
{
```

```
    // Scope 1
    // Variabel `a` dapat diakses disini

    let b = 2;
    var c = 3;
    const d = 4;

    // Variabel `b`, `c`, `d` dapat diakses
    {
        // Scope 2
        // Variabel `b`, `c`, `d` dapat diakses

        let e = 5;

        // Variabel `e` dapat diakses disini
    }

    // Variabel `e` tidak dapat diakses lagi
}

// Variabel `a`, `c` dapat diakses disini
// Variabel `b`, `d`, `e` tidak dapat diakses disini
```

Array

Array merupakan suatu kumpulan data yang sejajar di blok memori, array ini digunakan untuk menyimpan banyak data dan biasanya memiliki hubungan dengan yang lain seperti kumpulan data siswa dll, konsep Array ini dibawa di banyak bahasa pemrograman yan high level jadi prinsipnya akan sama.

Di javascript array ini dapat disimpan dengan variabel `const` karena array merupakan object, dan biasanya object jarang di definisikan ulang, tetapi untuk menambah data atau mengganti data bisa menggunakan [], Array memiliki index, index ini berupa biasanya berupa angka jadi ketika menggunakan [] kita harus memberikan index yang cocok, index di pemrograman mulai dari 0 kecuali bahasa pemrograman Lua yang mulai dari 1.

```
    // Index    0        1       2         3          4
const Siswa = ['Zagar', 'Akip', 'Alpin', 'Alifah', 'Syahrial'];

// Menganti data index ke 1
Siswa[1] = 'Ghaza';

// Menprint keseluruhan Array
console.log(Siswa);

// Menprint isi dari index ke 0 di Array Siswa
console.log(Siswa);
```

Control Flow

Merupakan suatu cara untuk mengendalikan suatu alur program, ini menggunakan data yang ada di dalam program tersebut dalam konteks ini Javascript untuk melakukan suatu operasi perbandingan yang akan menentukan block kode dijalankan atau tidak, Control Flow ada 4 jenis, `If statement`, `switch statement`, `While loop`, `For loop`, `Iterator/Foreach loop`, `Try Catch`, di jenis yang memiliki kata loop ini dapat mengulang kode yang sama berkali kali sampai operasi perbandingan tidak cocok lagi.

If

Merupakan suatu cara untuk mengatur alur program dengan melakukan perbandingan agar suatu block kode dijalankan atau tidak, `if` ini sebenarnya hanya melihat tipe data `Boolean` true berarti block kode jalan dan tipe data `Boolean` false tidak jalan, tetapi karena ada banyak tipe data jadi ada banyak cara untuk membuat hasil dari perbandingan menjadi tipe data `Boolean` seperti lebih besar atau lebih kecil atau mungkin mengecek apakah data di dalam variabel `null` atau tidak If juga dapat menjalankan kode ketika `Boolean` false.

```
// Selalu jalan
if(true) {
    document.write('Selalu jalan');
}else {
    // Kalau semua operasi gagal
    document.write('Tidak akan jalan');
}

// Selalu gagal
if(false) {
    document.write('Ini tidak akan jalan');
} else {
    // Karena operasi gagal ini jalan
    document.write('Ini akan selalu jalan');
}
```

`if` ini juga memiliki `else if`, ini digunakan untuk melakukan banyak perbandingan yang harus menjalankan kode yang berbeda sesuai dengan yang diberi ke if misal

```
let hari = 'Rabu';

// Melakukan perbandingan variabel hari dengan if statement, karena isinya `Rabu`
jadi block kode yang ada `hari == Rabu` akan dijalankan.
if (hari == 'Senin') {
    document.write('Upacara');
} else if (hari == 'Selasa') {
    document.write('Ulangan Harian MTK');
} else if (hari == 'Rabu') {
    // Melakukan sesuatu
} else if (hari == 'Kamis') {
    // Melakukan sesuatu
} else if (hari == "Jum'at") {
    // Melakukan sesuatu
} else if (hari == 'Sabtu') {
```

```
    // Melakukan sesuatu
} else if (hari == 'Minggu') {
    // Melakukan sesuatu
}
```

Kalau dilihat kurang rapi karena `if` tidak sering dipakai seperti ini, karena ini tugasnya cocok untuk si `switch`.

Switch

Sama dengan tetapi ini hanya menjalankan 1 perbandingan yaitu persamaan jadi kalau mau hal yang kompleks bisa kembali ke `if`

```
let hari = 'Rabu';

switch (hari) {
    case "Senin":
        // Kode
        break;
    case "Selasa":
        // Kode
        break;
    case "Rabu":
        // Kode
        break;
    case "Kamis":
        // Kode
        break;
    case "Jum'at":
        // Kode
        break;
    case "Sabtu":
        // Kode
        break;
    case "Minggu":
        // Kode
        break;
    default:
        // Kalau di-atas tidak ada yang cocok ini jalan
        break
}
```

While loop

While loop ini sama dengan yang if tetapi tidak ada block else-nya dan akan menjalankan block kode tersebut hingga perbandingannya memberikan `Boolean` false

```
let i = 0;

// Ini akan menjalankan block kode tersebut sampai variabel i tidak lagi dibawah 5
```

```
while (i < 5) {
    // Melakukan kegiatan

    i++; // Melakukan penjumlahan i = i + 1
}
```

For loop

Hampir mirip dengan `while` loop tetapi kita tidak harus mendefinisikan variabel `i` diluar (kalau tidak mau melakukan operasi yang sama untuk selamanya).

```
for(let i = 0; i < 5; i++) {
    // Melakukan kegiatan
}
```

Iterator / Foreach loop

Hampir mirip dengan `for` loop tetapi ini hanya dapat jalan dengan tipe data yang menimplementasikan Iterator seperti Array.

```
const myArray = ["Zagar", "Alpin", "Alifah", "Ghaza"];

// Bisa const atau let tergantung dengan mau diapakan
// value merupakan representasikan isi
// dan indek merupakan representasi posisi di dalam array
for (const [index, value] of myArray.entries()) {
    console.log(`${index} : ${value}`);
}

for (const value of myArray) {
    console.log(value)
}
```

Try Catch

Try Catch ini digunakan ketika kode kita mengalami `Exception` atau `Error`, kita akan dapat menjalankan block kode ketika sesuatu terjadi kesalahan, fitur ini dapat diterapkan dimana saja asalkan ada yang membuat error didalam block `try`, ketika error block `catch` akan dijalankan.

```
try {
    document.write("Melakukan proses...") // Ini akan dijalankan

    // Ketika sampai baris ini akan error karena kita membuat error dengan throw
Error
    throw Error("Sesuatu ada yang salah!");
}catch (e) {
```

```
    // Kalau error akan menjalankan block catch
    document.write("Ada error!");
    console.error(e);
}
```