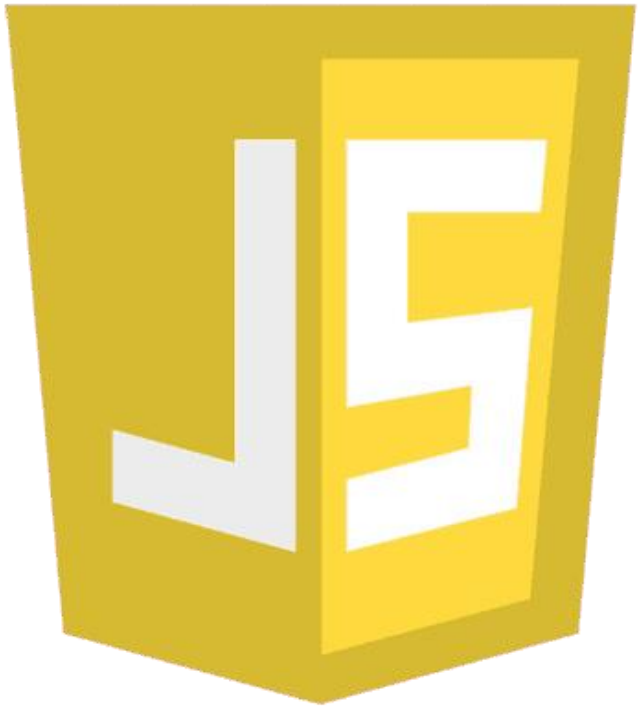

Javascript Optimization

Pertemuan 25-26

Created by : aninur.h@gmail.com

Javascript Optimization

- ✓ optimasi pada javascript berfungsi supaya waktu eksekusi program lebih singkat sehingga beban yang diperlukan untuk memproses program akan lebih ringan.

Created by : aninur.h@gmail.com

Beberapa hal dapat dilakukan untuk mengoptimasi :

✓ Mengurangi Aktivitas pada Loop

Looping banyak digunakan dalam pembuatan program. Setiap *statement* dalam dalam satu *loop*, termasuk *statement* for, dijalankan pada setiap iterasi *loop*. *Statement* yang dapat ditempatkan diluar *loop* akan membuat *loop* berjalan lebih cepat, contohnya seperti berikut:

```
//contoh pertama
let arr = [1, 2, 3];
let i;
for (i = 0; i < arr.length; i++) {}
```

```
//contoh kedua (optimasi)
let arr = [1, 2, 3];
let l = arr.length;
for (let i = 0; i < l; i++) {}
```

Beberapa hal dapat dilakukan untuk mengoptimasi :

✓ Mengurangi Akses DOM

Mengakses DOM pada HTML akan membuat proses eksekusi menjadi lebih lambat dibandingkan dengan akses *statement* Javascript lainnya. Jika ingin mengakses elemen DOM beberapa kali, akses satu kali dan simpan dalam sebuah *variable* local.

```
let obj;
obj = document.getElementById("demo");
obj.innerHTML = "Hello";
```

Dengan menyimpan DOM kedalam *variable*, kita dapat menggunakannya berkali kali melalui *variable* yang telah dibuat tanpa perlu mengakses DOM kembali .

Beberapa hal dapat dilakukan untuk mengoptimasi :

✓ Menghindari Variable Tidak Perlu

Dibeberapa kasus secara kita tidak sadari, kita membuat *variable* tertentu namun sebenarnya kita tidak membutuhkan *variable* tersebut. Seperti contoh berikut:

```
let fullName = firstName + " " + lastName;
document.getElementById("demo").innerHTML = fullName;
```

```
document.getElementById("demo").innerHTML = firstName + " " + lastName;
```

Beberapa hal dapat dilakukan untuk mengoptimasi :

```
const hello = () => {
return "hello";
};
```

Untuk memanggil *function* dan mendapatkan nilai kembali dari *function* tersebut, kita dapat melakukannya dengan cara berikut:

```
let i = hello();
console.log(i);
```


```
console.log(hello());
```

Beberapa hal dapat dilakukan untuk mengoptimasi :

✓ Menggunakan Array Helper Methods

Array helper methods banyak digunakan oleh *developer*. Hal ini sangat membantu *developer* sehingga tidak perlu membuat *loop* atau perulangan lagi. Beberapa *array helper methods* sebagai berikut:

forEach()

Method forEach() merupakan sebuah *method* yang digunakan untuk mengkases nilai yang ada dalam *array*, sama halnya seperti membuat *looping* untuk mengakses nilai dari sebuah *array*.

```
let arr = [1, 2, 3, 4, 5];
arr.forEach((item) => {
   console.log(item); // 1 2 3 4 5
});
```

Beberapa hal dapat dilakukan untuk mengoptimasi :

map()

Method map() ini pada dasarnya mirip dengan method forEach(), namun bedanya method map() ini dapat mengembalikan sebuah nilai .

```
let arr = [1, 2, 3, 4, 5];
let multiple = arr.map((item) => {
    return item * 2;
});
console.log(multiple); // [2, 4, 6, 8, 10]
```

Beberapa hal dapat dilakukan untuk mengoptimasi :

filter()

Fungsi *method* filter() adalah *method* yang digunakan untuk mempermudah dalam menyaring sebuah data.

```
let arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let mod = arr.filter((item) => {
return item % 2 == 0;
});
console.log(mod); // [2, 4, 6, 8, 10]
```

Beberapa hal dapat dilakukan untuk mengoptimasi :

find()

Sama halnya dengan method filter() , method find() ini juga untuk menyaring data, perbedaanya pada method find() ini hanya akan mengembalikan data pertama yang sesuai dengan kondisinya, jikalau method filter() akan melakukan penyaringan terhadap semua data.

```
let arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let mod = arr.find((item) => {
return item % 2 == 0;
});
console.log(mod); // [2, 4, 6, 8, 10]
```