Full Stack Development
sesi 16

Modern JS+-ES6

Sesi 16 | Modern JS - ES6

[Variable Declaration]

Di JavaScript ada dua keyword atau kata kunci yang bisa kita gunain untuk mendeklarasikan variable, yaitu: let, dan const. Sebenarnya ada satu lagi keyword yang bisa digunakan untuk mendeklarasikan variable, namanya var. Tapi untuk sekarang kita belum perlu sih pake var. Kita hanya belajar mendeklarasikan variable dengan let dan const saja.
Cukup tahu dulu aja kalau kita juga bisa menggunakan var.

Oke, saatnya kita melihat bagaimana cara menulis kode di JavaScript untuk mendeklarasikan variable.

Deklarasi Variabel dengan let

```
1  let playerName = "Budi";
2  console.log(playerName);
3  playerName = "Rudi";
4  console.log(playerName);
```

[Variable Declaration]

Memberi Nama Variabel

Jadi, di JavaScript dan bahasa programming pada umumnya ada istilah yang dinamakan dengan "reserved keyword", atau kata kunci yang sudah "dibooking" oleh bahasa pemrograman kita. Contoh dari reserved word adalah let. Kita tentu tidak bisa menggunakan let sebagai nama variable kita. Kalo kita buat nama variable let, browser akan marah-marah sama kita :)

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar#Keywords

[Data Types]

Jenis value ini di dunia pemrograman kita sebut dengan tipe data atau data types. Di JavaScript tipe data yang paling umum digunakan adalah:

- number,
- string,
- dan boolean

Berkenalan dengan tipe data NUMBER

Berkenalan dengan tipe data STRING

Berkenalan dengan tipe data BOOLEAN

Sesi 16 | Modern JS - ES6

[Data Types]

Teknik manipulasi tipe data STRING

- menggabungkan string - dengan ‘+’ sederhana

```
console.log("I" + " love " + "coding")
```

- menggabungkan string - dengan ‘+’ dalam beberapa baris

```
let kalimat = "This"
kalimat = kalimat + " is "
kalimat += "Javascript"
console.log(kalimat)
```

- menggabungkan string - dengan ` (backtick)

```
let first = "Modern"
let second = "Javascript"
console.log(`${first} ${second}`)
```

Sesi 16 | Modern JS - ES6

[Array]

Array adalah tipe data terstruktur yang berfungsi untuk menyimpan sejumlah data yang tipenya sama. Bagian yang menyusun **array** disebut dengan isi atau elemen **array**, masing-masing elemen bisa diakses melalui indeks **array**

-- Deklarasi array

```
let arr_bil = [1, 2, 3, 4, 5]
let arr_str = ["a", "buku", "bisa juga kalimat"]
let arr_bol = [true, false, false, true]
```

-- Operasi array - PUSH

```
let arr_bil = [10, 29, 2, 3]
console.log(arr_bil)
arr_bil.push(24)
console.log(arr_bil)
```

Sesi 16 | Modern JS - ES6

[Array]

-- Operasi array - SHIFT

```
1  const array1 = [1, 2, 3];
2  const firstElement = array1.shift();
3  console.log(array1);
4  console.log(firstElement);
```

-- Operasi array - UNSHIFT

```
1  const array1 = [1, 2, 3];
2  console.log(array1.unshift(4, 5));
3  console.log(array1);
```

-- Operasi array - SPLICE

```
1  const months = ['Jan', 'March', 'April', 'June'];
2  months.splice(1, 0, 'Feb');
3  console.log(months);
4  months.splice(4, 1, 'May');
5  console.log(months);
```

[Array Multidimensi]

-- Deklarasi array

```
let arr = [
  [1, 2, 3],
  [7, 3],
  [91, 8, 100, 30]
]
console.log(arr)
```

-- Operasi - operasi pada array multidimensi adalah sama dengan operasi pada array 1 dimensi, hanya saja kita perlu benar-benar memperhatikan index array yang di operasikan

```
let arr = [
  [1, 2, 3],
  [7, 3],
  [91, 8, 100, 30]
]
console.log(arr)

arr[0].shift()
console.log(arr)

arr[1].unshift(4, 5)
console.log(arr)
```

Sesi 16 | Modern JS - ES6

[this Keyword]

this di JavaScript sebenarnya sedikit berbeda dibandingkan dengan bahasa pemrograman lain. this memiliki beberapa aturan. Aturan pertama, thisseringkali merupakan variabel global. Contohnya jika di front-end atau di browser this
dapat merupakan window jika context-nya global.

Atau jika kita eksekusi this di terminal via node REPL, this adalah global variable dari Node.js.

Aturan kedua, jika this berada dalam konteks sebuah object, this dipakai untuk memanggil properties yang dipunyai object tersebut.

Aturan ketiga, dalam konteks object-oriented di JavaScript yang akan kita pelajari nanti di bagian-bagian berikutnya, this digunakan untuk mendeklarasikan public properties.

```
1  let counter = {
2    val: 0,
3    increment: function() {
4      this.val += 1
5    }
6  }
```

Sesi 16 | Modern JS - ES6

[Arrow Function]

Ada beberapa keuntungan menggunakan arrow function

1. Sintaks jadi lebih pendek,
2. Implicit Return,
3. Memudahkan memahami this keyword

```
// Traditional Function (no arguments)
let a = 4;
let b = 2;
function (){
  return a + b + 100;
}

// Arrow Function (no arguments)
let a = 4;
let b = 2;
() => a + b + 100;
```

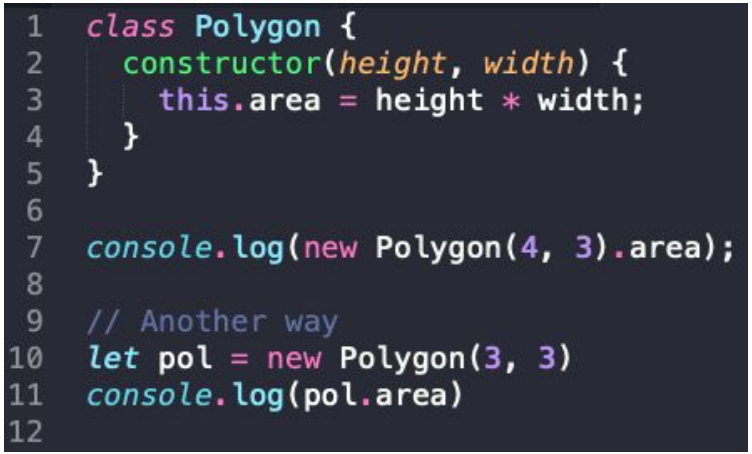
```
// Traditional Function
function (a){
  return a + 100;
}

// Arrow Function

// Cara 1
(a) => {
  return a + 100;
}

// Cara 2
(a) => a + 100;

// Cara 3
a => a + 100;
```

```
// Named function, stored as value
const tambah = (a, b) => a + b
```

Sesi 16 | Modern JS - ES6

[Class]

Selain class-class yang sudah disediakan oleh JavaScript, sekarang kita pun bisa mendefinisikan class baru, sesuai dengan kebutuhan kita



```
1  class Polygon {
2    constructor(height, width) {
3      this.area = height * width;
4    }
5  }
6
7  console.log(new Polygon(4, 3).area);
8
9  // Another way
10 let pol = new Polygon(3, 3)
11 console.log(pol.area)
12
```

Destructuring Object

Sesi 16 | Destructuring Object

[Review Object Literal]

So, masih inget ga guys dengan Object Literal ? Kita refresh sedikit yuks, sebelum kita masuk ke materi yang lebih dalam soal object ini

```
// Deklarasi cara 1
let obj1 = {
  nama: "Agnes",
  umur: 20
}

// Deklarasi cara 2
let obj2 = {}
obj2.nama = "Flavia"
obj2.umur: 25
```

```
// Menampilkan object
console.log(obj1)
console.log(obj2)

// Akses Object
console.log(obj1.nama)
console.log(obj2.umur)
```

Sesi 16 | Destructuring Object

[Destructuring Object]

ES6 memperkenalkan fitur baru yang namanya destructuring. Untuk memahami apa itu destructuring, mari kita review bagaimana penggunaan Object Literal di JavaScript. Untuk menambahkan properti baru dalam object, kita menggunakan dot notation.

```
1  const user = {};
2  user.name = 'Adi Nugroho';
3  user.handle = '@adinugroho';
4  user.location = 'Jakarta, Indonesia';
```

Dengan menggunakan dot notation seperti disamping kita dipaksa harus menambahkan properti satu per satu. Dot notation juga dapat kita gunakan untuk mendapatkan data dari object literal atau istilah kerennya ekstraksi data.

```
1  const user = {};
2  user.name = 'Adi Nugroho';
3  user.handle = '@adinugroho';
4  user.location = 'Jakarta, Indonesia';
5
6  // extraction
7  const name = user.name;
8  const handle = user.handle;
9  const location = user.location;
10 console.log(name, handle);
```

Sesi 16 | Destructuring Object

[Destructuring Object]

Sekarang, dengan adanya destructuring object, kita dapat menghemat waktu dan kode dengan menuliskan lebih sedikit baris kode, terutama untuk extraction. Tapi, tentu saja, kecermatan dan ketelitian tetap harus temen-temen perhatikan

```
1  const user = {};
2  user.name = 'Adi Nugroho';
3  user.handle = '@adinugroho';
4  user.location = 'Jakarta, Indonesia';
5
6  // extraction
7  const { name, handle, location } = user;
8  console.log(name, handle, location);
```

Sesi 16 | Destructuring Object

[Destructuring Function Result]

Lanjut, kita pun dapat melakukan destructuring terhadap *return value* dari sebuah *function* yang mengembalikan nilai dalam bentuk object. WOW...

```
1  const getUser = () => {
2    return {
3      name: 'Adi Nugroho',
4      handle: '@adinugroho',
5      location: 'Jakarta, Indonesia'
6    };
7  }
8
9  // extraction
10 const { name, handle, location } = getUser();
11 console.log(name, handle, location);
```

Sesi 16 | Destructuring Object

[Destructuring Array]

Ada kabar baik, Array pun bisa kita destructuring. Meskipun tidak banyak kasus yang membutuhkannya, destructuring bisa membantu banyak dalam hal ini. Hanya saja, kita tidak menggunakan kurung kurawal ({ }) , tapi kita menggunakan kurung siku ([ ])

```
1  let spec = ["Mercedes Benz", "Mercy", "GLA 200"]
2
3  let [brand, short_name, type] = spec
4  console.log(brand, short_name, type)
```