Front End Bootcamp
PT. Mahardika Solusi Teknologi

Ariel Arliyanus, S.Kom

Chapter 10 – JavaScript ES6

Chapter 10
JavaScript ES6

ES6 atau EcmaScript 6 atau EcmaScript 2015 merupakan fitur bahasa pada pemrograman Javascript modern atau biasa di sebut next gen javascript. lalu kenapa kita harus mempelajari ES6, itu karena di reactJS kita akan banyak menggunakan sintaks-sintaks ES6.

Daftar Fitur ES6

- Arrows Function
- Classes
- Enchanced object literals
- Template strings
- Destructuring
- Default + rest + spread
- Let + const
- Iterators + for..of
- Generators
- Unicode
- Modules
- Modules loaders
- Map +set + weakmap + weakset
- Proxies
- Symbols
- Subclassable built-ins
- Promises
- Math + number + string + array + object apis
- Binary dan octal literals
- Reflect API
- Tail calls

Tidak perlu menguasai seluruh fitur diatas, tetapi kita akan membagi fitur-fitur yang penting untuk di pelajari yaitu:

- let + const
- arrow function
- default paramater
- Template literal
- Enhanced object literals
- Destructuring
- Rest Parameters + Spread Operator
- promise (nanti ada materinya di asynchronous)
- class

Let + Const

let dan const merupakan statement untuk mendefinisikan variable sama seperti var hanya saja terdapat perbedaan diantara let + const dan var.

let hampir sama seperti var hanya saja jika dalam satu block kode ada sebuah let yang sudah di definisikan kita tidak dapat mendefinisikan kembali kecuali kita membuat block kode baru di dalamnya misal seperti menambahkan if

sedangkan const sendiri bersifat tidak bisa di rubah seperti sebuah konstanta dalam matematika

var sendiri masih bisa digunakan untuk menjaga compability dengan versi sebelumnya

berikut ini perbandingan var dengan let + const:

Normal Javascript:

```
var x = 1;

if (x === 1) {
var x = 2;

console.log(x);
// expected output: 2
}

console.log(x); // 2
```

ES6 :

```
let x = 1;

if (x === 1) {
  let x = 2;

  console.log(x);
  // expected output: 2
}

console.log(x); // 1

const number = 42;
number = 100; // Uncaught TypeError: Assignment to constant variable.
```

Arrow Functions

arrow functions merupakan fitur yang ada pada es6 bisa dibilang lebih singkat dari function biasa, function biasa sendiri masih bisa di gunakan.

berikut ini perbandingan penggunaan function dan arrow function:

Normal Javascript:

```
function myFunction (){
    // isi Function
}
// panggil Function
myFunction()
```

ES6 :

```
const myFuncton = () => {
    //function
}

// panggil Function
myFunction()
```

Default Parameters

biasanya kita dalam membuat function pasti punya parameter tapi apakah parameter itu sendiri bisa di beri default, jawabannya adalah ya di ES6 kita dapat memasukkan default parameter seperti contoh di bawah ini:

```
function multiply(a, b = 1) {
  return a * b;
}

console.log(multiply(5, 2));
// expected output: 10

console.log(multiply(5));
// expected output: 5
```

Template Literals

template literal atau biasa di sebut template string merupakan fitur ES6 yang memungkinkan kita menyusun string dengan rapi dengan menggunakan tanda petik terbalik dan ${variabelnya}.

berikut ini cara menggunakan template literal:

```
const firstName= "John"
const lastName = "Doe"
const teamName = "Mr"

const theString = `${firstName} ${lastName}, ${teamName}`

console.log(theString) // John Doe, Mr
```

Enhanced object literals

Enhanced object literals merupakan fitur ES6 yang memungkinkan kita untuk menyederhanakan sebuah object, dimana biasanya kita selalu menulis property dan value, tetapi jika terdapat kondisi ada variabel yang namanya sama dengan property maka kita bisa assign hal tersebut sebagai value tetapi dengan hanya menulis property nya saja

berikut ini contoh enhanced object literals dan perbandingan dengan javascript sebelum es6

Before ES6 Javascript:

```
const fullName = 'John Doe'

const john = {
  fullName: fullName
}
```

After ES6 Javascript:

```
const fullName = 'John Doe'

const john = {fullName}
```

Destructuring

Destructuring merupakan ekspresi javascript yang memungkinkan untuk membagi atau memecah nilai dari sebuah array atau objek ke dalam variabel yang berbeda

berikut ini contoh penggunaan destructuring dan perbandingannya dengan sebelum destructuring

tanpa destructuring:

```
// array
var numbers = [1,2,3]

var numberOne = numbers[0]
var numberTwo = numbers[1]
var numberThree = numbers[2]

console.log(numberOne)

// object
var studentName = {
    firstName: 'Peter',
    lastName: 'Parker'
};

const firstName = studentName.firstName;
const lastName = studentName.lastName;

console.log(firstName)
```

dengan destructuring:

```
// array
let numbers = [1,2,3]

const [numberOne, numberTwo, numberThree] = numbers

console.log(numberOne)

// object
var studentName = {
    firstName: 'Peter',
    lastName: 'Parker'
};

const {firstName, lastName} = studentName

console.log(firstName)
```

Rest Parameters + Spread Operator

Rest Parameters dan Spread Operator di lambangkan dengan simbol yang sama yaitu "..."

Rest Parameters

Rest Parameter ini berguna untuk menggabungkan semua paramater pada function ke dalam array. Dengan menggunakan Rest Parameter ini dapat membantu kita mendefinisikan function dengan rapi serta memberikan parameter yang tidak terbatas pada sebuah function.

berikut ini contoh penggunaan rest parameters:

```
// Rest Parameters

//first example
let scores = ['98', '95', '93', '90', '87', '85']
let [first, second, third, ...restOfScores] = scores;

console.log(first) // 98
console.log(second) // 95
console.log(third) // 93
console.log(restOfScores) // [90, 87, 85]

//second example
const filter = (...rest) =>{
return rest.filter(el => el.text !== undefined)
}

console.log(filter(1, {text: "wonderful"}, "next")) // wonderful

//third example
const fullName = (...rest) =>{
let [firstName, lastName] = rest
return `${firstName} ${lastName}`
}

console.log(fullName("John", "Doe")) // John Doe
```

Spread Operator

Spread Operator digunakan untuk membagi elemen array atau properti pada objek, sehingga elemen array dapat ditambahkan/dimasukan ke dalam array baru

berikut ini contoh penggunaan Spread Operator:

```
// spread operator
let array1 = ['one', 'two']
let array2 = ['three', 'four']
let array3 = ['five', 'six']

// ES5 Way / Normal Javascript

var combinedArray = array1.concat(array2).concat(array3)
console.log(combinedArray) // ['one', 'two', 'three', 'four', 'five', 'six']

// ES6 Way

let combinedArray = [...array1, ...array2, ...array3]
console.log(combinedArray) // ['one', 'two', 'three', 'four', 'five', 'six']

//Spread in object

let person = {name: "john", age: 30}

let newPerson = {...person, hobby: "Gaming"}

console.log(newPerson) // {name: "john", age: 30, hobby: "Gaming"}
```