8. Fundamentals Javascript

28 March 2022 6:17

Variables

Terdapat 3 cara deklarasi variable di javascript. Yaitu dengan :

- var
- let
- const

Ketiganya memiliki karakteristik masing-masing. Hal ini penting untuk diketahui agar program yang kita buat terhindar dari *bug*. Tabel berikut akan memvisualisasikan karakteristik ketiga jenis variabel tersebut :

Jenis	Redeclare	Hoisting	Block Scope
var	✅	✅	❌
let	❌	❌	✅
const	❌	❌	✅

var

Mendeklarasikan variabel menggunakan var merupakan cara lama di javascript. Karena penggunaanya yang dinilai rawan menimbulkan *bug*. Namun sebelum kita mengetahui lebih lanjut mengapa var dapat menimbulkan bug, simak karakteristiknya terlebih dahulu :

Karakteristik 1 (Redeclare) : var dapat diinisialisasi kembali

```
var name = "Arief";
var name = "Kurniawan";
console.log(name); // Kurniawan
```

Karakteristik 2 (Hoisting) : var bersifat hoisting

```
x = 5;
var x;
console.log(x);
```

Karakteristik 3 (Block Scoped)* : var tidak memiliki *block scoped

```
var dog = 'Ralf';
if (true) {
  var dog = 'Skip';
}
```

```
console.log(dog); // Skip
```

let

deklarasi variable menggunakan let adalah cara baru yang diperkenalkan oleh javascript di es6. let adalah cara yang banyak dipakai oleh saat ini. karena kemampuannya dan juga mendukung *function* scoped, berikut beberapa karakteristik dari let :

Karakteristik 1 (Redeclare) : let tidak bisa diinisialisasi kembali

```
let x = 1;
let x = 2;
console.log(x); // SyntaxError: Identifier 'x' has already been declared
```

namun, nilai let bisa di ubah kembali, contohnya sebagai berikut :

```
let x = 1;
x = 2;
console.log(x); // 2
```

Karakteristik 2 (Hoisting) : let tidak bisa hoisting

```
x = 5;
let x;
console.log(x); // ReferenceError: Cannot access 'x' before initialization
```

Karakteristik 3 (Block Scoped) : let memiliki block scoped

```
let dog = 'Ralf';
if (true) {
  let dog = 'Skip';
}
console.log(dog); // Ralf
```

const

deklarasi variable menggunakan const juga merupakan cara baru di es6. penggunaan const lebih ketat dari let maupun var.

Karakteristik 1 (Redeclare) :
const tidak dapat diinisialisasi ulang

```
const x = 1;
const x = 2;
console.log(x); // SyntaxError: Identifier 'x' has already been declared
```

const tidak dapat diubah nilainya

```
const x = 1;
x = 2;
console.log(x); // TypeError: Assignment to constant variable.
```

gunakanlah const ketika membutuhkan sebuah variabel yang nilainya tidak akan diubah kembali. Namun, alangkah lebih baik kita selalu menggunakan const setiap kita membuat variable di javascript. berikut penggunaan umum dari const :

Karakteristik 2 (Hoisting) : const tidak bisa hoisting

```
x = 5;
const x;
console.log(x); // SyntaxError: Missing initializer in const declaration
```

Karakteristik 3 (Block Scoped) : const memiliki block scoped

```
const dog = 'Ralf';
if (true) {
  const dog = 'Skip';
}
console.log(dog); // Ralf
```

Perilaku const pada Array dan Object

```
const someArr = [3, 4, 5];
someArr.push(6);
console.log(someArr); // [3,4,5,6]
```

```
const someObj = {
  dog: 'Skip',
  cat: 'Caramel',
  bird: 'Jack'
};
someObj.camel = 'Bob';
console.log(someObj); //{ dog: 'Skip', cat: 'Caramel', bird: 'Jack',
camel: 'Bob' }
```

Data Types

Primitive Types

Numbers

```
1    age = 26;                    //number
2    weight = 156.7;              //floating point
```

Strings

```
1    firstName = "Mark";          //double quote
2    famillyName = 'Twain';       //single quote
```

Booleans

```
1    z = true;
2    y = false;
```

Special Data Types

null

```
1    age = null;    //was 26 before. null is an object
```

Undefined

```
1    address;              //no value
2    a = function() {};    //no return statement
```

Different

empty (undefined) variable declared by the name "box"

null is like a vacuum. It's an object, it's there, but it's nothing

Object data Types

Object

```
employee = {name:"Joe", age:21, id:99};
function x() {};
```

Array

```
name = ["Joe", "Mike", "Freddy"];
```

Operators

Selanjutnya, kita akan membahas mengenai operator Operator dalam bahasa pemrograman sendiri adalah simbol yang memberi tahu interpreter untuk melakukan operasi seperti matematika, relasional, atau logika untuk memberikan hasil tertentu.
 , ada 2 jenis operator : *Binary Operators* dan *Unary Operators* Apa ya perbedaanya ? . Kita lihat penjelasannya yuk.

Binary Operators

Arithmetic Operator

Operator ini digunakan untuk melakukan operasi matematika seperti : tambah, kali , bagi , kurang dan modulus terhadap angka. Berikut simbol dan implementasi nya dalam program :

```
+, -, *, /, %
```

```
let a = 10;
let b = 2;
let c = a+b;
console.log(c)  // Coba operator lain
```

Assignment Operator

operator ini adalah tanda sama dengan (=), di mana tanda ini digunakan untuk

menginisialisasi nilai pada variabel. Tempatkan variabel yang ingin diberi nilai di sebelah kiri, sementara nilainya di sebelah kanan. Di antara keduanya terdapat operator assignment.

=, *=, /=, %=, +=,-=,

```
let a = 10;
let b = 2;
a += b ;
console.log(a) // sama dengan a = a + b; coba operator lain
```

Comparison Operator

operator komparasi sebagai logika dasar dalam membandingkan nilai pada JavaScript. Terdapat serangkaian karakter khusus yang disebut dengan operator pembanding/komparasi yang dapat mengevaluasi dan membandingkan dua nilai. Berikut daftar operator dan fungsinya:

==, !=, ===, !==, >, >=, <, <=

Berikut penjelasannya mengenai tiap-tiap operatornya :

Operator	Fungsi
<	Membandingkan dua buah operand apakah nilai pertama lebih kecil dari nilai kedua
<=	Membandingkan dua buah operand apakah nilai pertama lebih kecil atau sama dengan dari nilai kedua
>	Membandingkan dua buah operand apakah nilai pertama lebih besar dari nilai kedua
>=	Membandingkan dua buah operand apakah nilai pertama lebih besar atau sama dengan dari nilai kedua
!=	Membandingkan apakah kedua operand apakah Tidak sama (tidak identik)

==	Membandingkan apakah kedua operand apakah sama (tidak identik)
!==	Membandingkan kedua nilai apakah tidak identik
===	Membandingkan kedua nilai apakah identik

Code :

```
let a = 10;
let b = 2;
console.log(a < b) // true
console.log(a != b) // true
```

Setiap kita bandingkan maka akan menghasilkan nilai boolean. Dalam javascript ada perbandingan nilai yang identik dan tidak identik, apa kedua hal itu ?. Berikut contohnya :

```
let aNumber = 10;
let aString = '10';
console.log(aNumber == aString) // true
```

Jika kita menggunakan == (dua sama dengan) operator. Maka hal ini akan terlihat sama sehingga tidak memedulikan tipe data dan hasilnya akan true.

```
let aNumber = 10;
let aString = '10';
console.log(aNumber === aString) // false
```

Jika kita menggunakan === (tiga sama dengan) operator. Maka hal ini akan terlihat berbeda dan sangat memperhatikan tipe data dan hasilnya akan false.

Logical Operator

operator lain yang dapat kita gunakan untuk menetapkan logika yang lebih kompleks, yakni dengan logical operators. Dengan logical operator, kita dapat menggunakan kombinasi dari dua nilai boolean atau bahkan lebih dalam menetapkan logika.

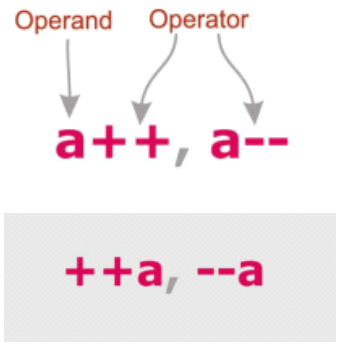
&&, ||, !

```
let a = 10;
```

```
let b = 12;
// AND operator
console.log(a < 15 && b > 10); // true && true = true
console.log(a > 15 && b > 10); // false && true = false
// OR operator
console.log(a < 15 || b > 10); // true || true = true
console.log(a > 15 || b > 10); // false || true = true
// NOT operator
console.log(!(a < 15)); // !true = false
console.log(!(a < 15 && b > 10)); // !true && true = (true) false
```

Increment and Decrement

Increment artinya setiap nilai meningkat, *begitupun Decrement* artinya setiap nilai berkurang berikut ada dua tipe increment/decrement:



Postfix Increment

```
let x = 3;
y = x++;
console.log(y); // 3
```

Prefix Increment

```
let x = 3;
y = ++x;
console.log(y); // 4
```

Percabangan

If

```
const isRaining = true;
console.log("Persiapan sebelum berangkat kegiatan.");
if (isRaining) {
    console.log("Hari ini hujan. Bawa payung.");
}
console.log("Berangkat kegiatan.");
/* output:
Persiapan sebelum berangkat kegiatan.
Hari ini hujan. Bawa payung.
Berangkat kegiatan.
 */
```

else

```
let x = 50;
if(x > 70) {
console.log(x);
} else {
console.log("Nilai kurang dari 70");
}

/* output
Nilai kurang dari 70
*/
```

else if

```
let language = "French";
let greeting = "Selamat Pagi"
if(language === "English") {
greeting = "Good Morning!";
} else if(language === "French") {
greeting = "Bonjour!"
} else if(language === "Japanese") {
greeting = "Ohayou Gozaimasu!"
}
console.log(greeting);
/* output
Bonjour!
*/
```

Ternary Operator

```
// condition ? true expression : false expression
const isMember = false;
const discount = isMember ? 0.1 : 0;
console.log("Anda mendapatkan diskon sebesar " + discount * 100 + "%");
/* output
Anda mendapatkan diskon sebesar 0%
*/
```

Switch Case

```
let language = "French";
let greeting = null;
switch (language) {
    case "English":
        greeting = "Good Morning!";
        break;
    case "French":
        greeting = "Bonjour!";
        break;
    case "Japanese":
        greeting = "Ohayou Gozaimasu!";
        break;
    default:
        greeting = "Selamat Pagi!";
}
console.log(greeting);
/* output
Bonjour!
*/
```

Looping (Perulangan)
While

```
let i = 0;
let text = "";
while(i < 10) {
    text += `The number is ${i}`
    i++
```

```
    console.log(i);
}
```

do while

```
let i = 0;
let text = "";
do{
    text += `The number is ${i}`
    i++;
    console.log(i);
} while(i < 10);
```

Perbedaan while dan do while

While akan mengecek kondisi terlebih dahulu, jika false , maka badan perulangan tidak dimasuki

```
let i = 0;
let text = "";

while(i < 0) {
    text += `The number is ${i}`
    i++
    console.log(i);
}
// tidak ada output
```

Sementara itu , do while, dia masuk dulu ke dalam badan perulangannya. Artinya minimal dijalankan sekali. Jika kondisi false maka perulangan berhenti;

```
do{
    text += `The number is ${i}`
    i++;
    console.log(i);
} while(i < 10);

// print 1 .
```

For loop

```
let i = 0;

for(i = 0; i < 5; i++) {
    console.log(i);
}
```

Function Declaration

Fungsi merupakan blok program yang dibungkus dengan keyword function yang dimana jika kita ingin menjalankannya, maka kita harus memanggilnya terlebih dahulu. Fungsi ini bersifat reusable/ bisa dipakai berkali-kali.

```
function printHello() {
   console.log("Hello World");
}
```

```
// panggil fungsi
printHello()
```

Recursive

```
function factorial(x) {
    if (x <= 1) return 1;
    return x * factorial(x-1);
}
const result = factorial(10);
console.log(result);
```

Function Expression

Fungsi ini disimpan pada variable, sehingga jika memanggilnya, kita panggil nama variabel nya

```
const square = function(x) { return x*x};

// panggil fungsi
square(4);
```

Parameters

Merupakan varibel yang kita tetapkan pada fungsi. Untuk memanggilnya kita harus input sebuah nilai

```
const square = function(x)
```

Argument

Merupakan nilai yang kita masukkan kedalam fungsi yang memiliki parameter

```
square(4);
```

Spread Operator

Spread operator berfungsi menampung arguments berapapun yang dimasukkan kedalam fungsi

```
function sum(...theArgs) {
  return theArgs.reduce((previous, current) => {
    return previous + current;
  });
}
```

Default value

Default value merupakan nilai yang kita tetapkan pada parameter. Ketika kita memanggil fungsi dengan tanpa parameter. Maka defaultnya akan ditampilkan. Tapi jika , tetap memasukkan parameter. Maka nilainya akan di update berdasarkan input kita (ditimpa).

```
function greet(name = "Stranger"){
    console.log("Hello" + name);
}
greet();
greet("Arief");
```

Arrow Function

Di es6 javascript , perkenalkan , *arrow function* yang berguna untuk menyingkat function expression sehingga bisa satu baris

```
const sum = (x,y) => {
    return x + y;
}
```

Quiz :

Buatlah fungsi pangkat. Dengan satu parameter. Fungsinya mengembalikan perkalian pangkat. Misal. Saya input ke fungsi pangkat(3) . Maka nilainya akan 27.

Array

Array adalah kumpulan nilai yang terurut. Setiap nilai
disebut elemen, dan setiap elemen memiliki posisi numerik dalam array, yang dikenal sebagai indeksnya.

Definisi array

```
const arr = [1,2,3];
```

Akses index

```
console.log(arr[0]);
```

Add item to array

```
arr[2] = "Arief" // replace with the new one
arr[3] = "Kurniawan" // add item
```

Array Method (map, filter, reduce)

map

```
const arr = [1,2,3].map(x => x*x);
console.log(arr);
```

filter

```
const arr = [5,4,3,2,1].filter(x => x <= 3);
console.log(arr);
```

reduce

```
const arr = [5,4,3,2,1].reduce((x,y) => x + y);
console.log(arr);
```

Spread Operator for cloning arr

```
const arr = [5,4,3,2,1];
const arr2 = [...arr , null, 2,3,1];
console.log(arr2);
```

Quiz :
Isikan array dengan 20 item, lalu filter bilangan genap, selanjutnya tiap2 itemnya dikalikan 2. terakhir jumlahkan seluruh item.

Object

sebaliknya, objek digunakan untuk menyimpan koleksi kunci dari berbagai data dan lebih kompleks
entitas. Dalam JavaScript, objek menembus hampir setiap aspek bahasa. Jadi kita harus
memahaminya terlebih dahulu sebelum masuk lebih dalam ke tempat lain.
Sebuah objek dapat dibuat dengan kurung kurawal {…} dengan daftar properti opsional. Sebuah properti
adalah pasangan "kunci: nilai", di mana kunci adalah string (juga disebut "nama properti"), dan nilai dapat
menjadi apa saja.

Definisi

```
const apple = {
    name: 'Macintosh',
    color: 'Silver',
    weight: 100,
    display: function() {
        console.log("I weight" + this.weight + "g");
    }
}
```

Membaca (get) item

```
console.log(apple.name)
```

menambah (set) item

```
apple.belongsTo = "Arief"
```

Quiz

```
const user = {};
```

Isikan objek tersebut dengan ketentuan :
Nama depan, nama belakang, alamat, hoby, method perkenalan diri.