Statement (sakinys)

Expression (išraiška)

Operatoriai
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators

Duomenų tipai

- Number (skaičiai)
- String (eilutės/tekstas)
- Boolean (loginės)
- `Null` reikšmė (ši reikšmė yra be tipo)
- `Undefined` reikšmė (žymi, kad kintamasis yra neinicializuotas)
- Object (objektai, įskaitant ir masyvus)
- Function

Kintamųjų deklaravimas

Yra 4 būdai:

```
var a = 2;

let b = 2;
const c = 2;

function x() {}
```

Blokai

```
var a = 20;

{
     a = 10;
     foo(a / 2);
}

var c;
```

Vieni blokai iš esmės nieko neatlieka, tik vizualiai sugrupuojam kodą. Suveiks kiekvieną kartą.

Tačiau galima pridėti sąlygą / sakinį, kada tas blokas bus vykdomas. Jei sąlyga true, blokas bus vykdomas. Pridėti pavyzdį su while.

Blokai taip pat matomi su function statement. Blokas susiejamas tik su ta funkcija. Jis iškart nevykdomas, tik kai ta funkcija pakviečiama.

Falsy values

```
0
-0
NaN
“”
false
Null
undefined
```

For Loops
Initialization clause (sąlyga)
Conditional / test clause (sąlyga)
Update clause (sąlyga)

```
for (a = 5; a < 10; a = a +1) {
      console.log(a);
}

a = 5;
while (true) {
    if (a>= 10) {
        break;
    }
    console.log(a);
    a = a + 1;
}
```

Visos 3 dalys nėra būtinos. Jei: parašysim: `for (;;) {}`
Ciklas suksis be pabaigos. Todėl, kad nėra jokios sąlygos kuri bus lygi false, kuri pasakys, kad sustabdyti for ciklą.

Pvz:

```
var cities = ["Melbourne", "Amman", "Helsinki", "NYC", "Boston",
"Chicago"];

for (var i = 0; i < 6; i++) {
    console.log("I would like to visit " + cities[i]);
}
```

```
var coinFace = Math.floor(Math.random() * 2);

while(coinFace === 0){
      console.log("Heads! Flipping again...");
      var coinFace = Math.floor(Math.random() * 2);
}
console.log("Tails! Done flipping.");
```

Funkcijos
Būdas logiškai sugrupuoti kodą į vieną vietą ir pakviesti tą kodo dalį kai tau reikia. Panašu į ciklus, tačiau vykdomas tiek kartų kiek pakviečiam. Galbūt tik vieną kartą, ar daug kartų. Funkcijos gali grąžinti kažkokį rezultatą, gali negrąžinti.

Pavyzdys
Kai tik kažkas kartojasi, geriau iškelti į atskirą funkciją.

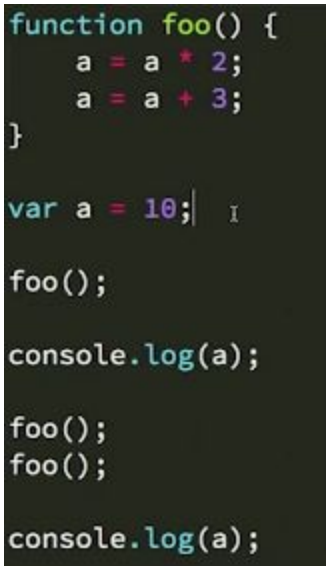
```
function printAmount() {
      console.log( amount.toFixed( 2 ) );
}

var amount = 99.99;

printAmount(); // "99.99"

amount = amount * 2;

printAmount(); // "199.98"
```



Funkcijos gali priimti parametrus. Funkcijos veikimas priklauso nuo paduotų parametrų. Parametrų skaičius neribotas.

Kai kviečiame (call) funkciją - paduodame argumentus.
Kai apibrėžiame (declare) - aprašome parametrus.

Funkcijai galima paduoti kitą funkciją. Skirtumą paaiškinti tarp:

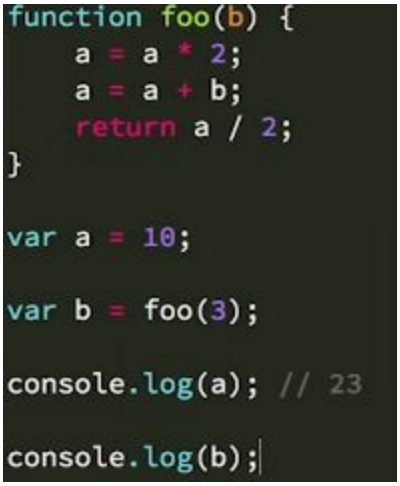
```
function foo(b) {
   console.log(typeof b);
}
function bar () {
    return 5;
}
foo( bar )
foo( bar() )
```

Scope

Šitam pavyzdyje b yra vidinis kintamasis. Galim deklaruoti trečią kintamąjį c viduje funkcijos. Jis egzistuos tik toje funkcijoje.

Scope - techninis terminas, galit įsivaizduoti kaip burbulus.

Svarbu: Visada gali pasiekti kintamuosius esančius aukštesniame scope (vienu lygiu aukščiau). Atvirkščiai yra netiesa.



```
function foo(b) {
    a = a * 2;
    a = a + b;
    return a / 2;
}

var a = 10;

var b = foo(3);

console.log(a); // 23

console.log(b);
```