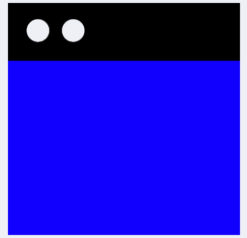
Code Academy



Gytis Juozenas

JavaScript kintamieji, duomenų tipai

2022

JavaScript programavimo kalba

Šiandien išmoksite

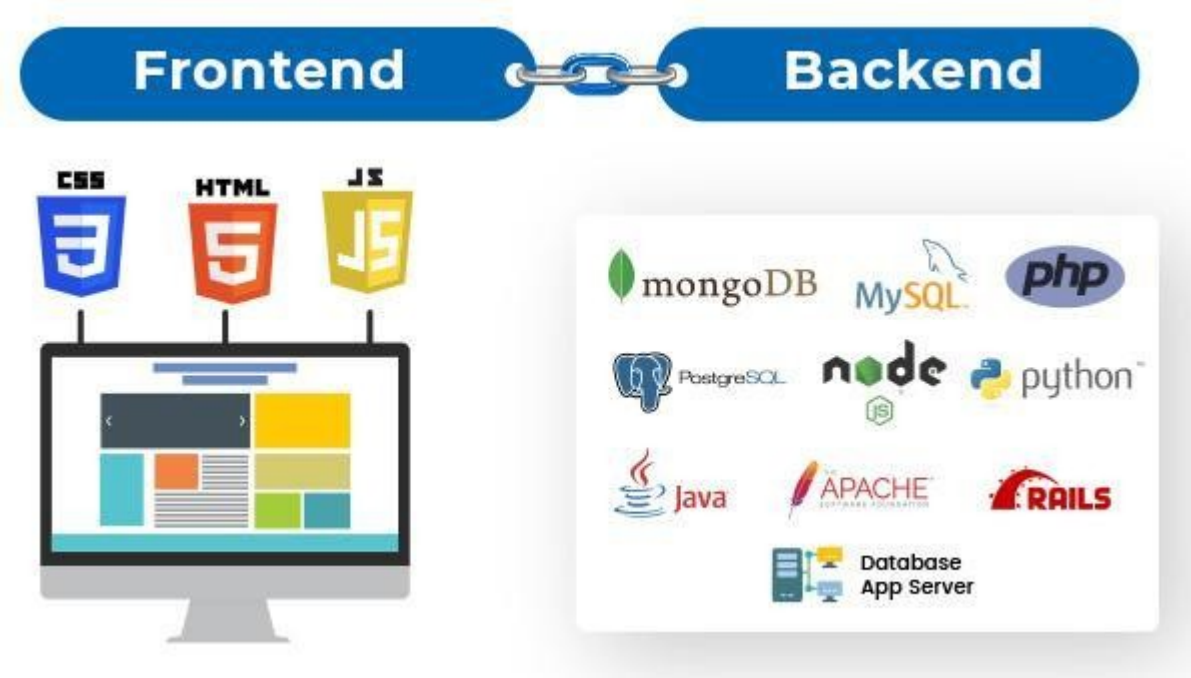
01 Kas yra JavaScript?

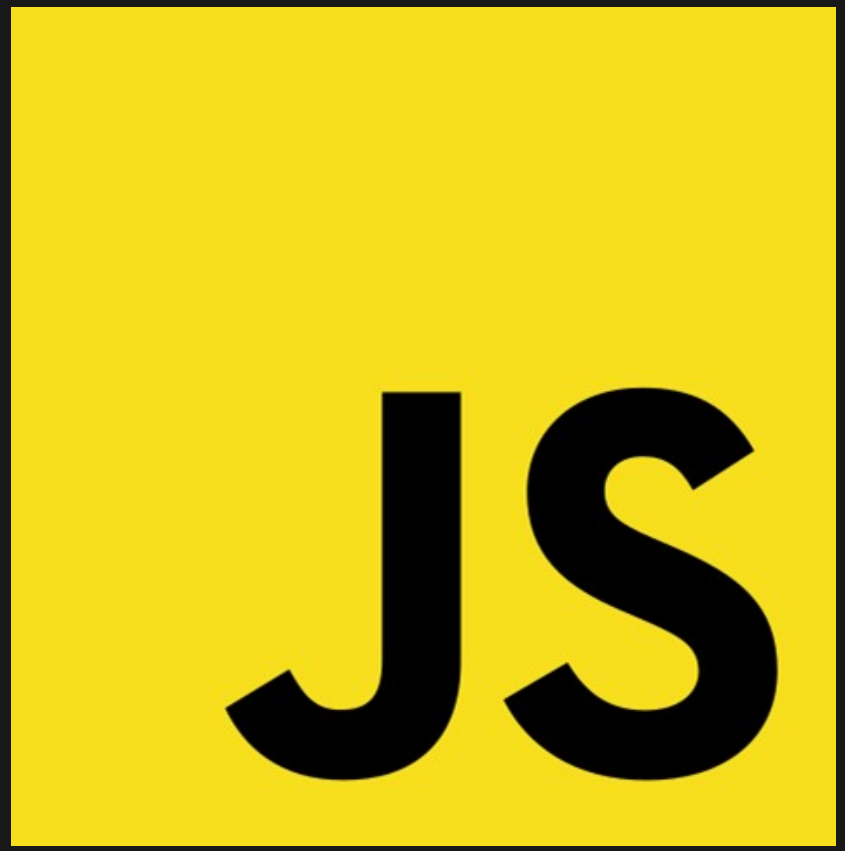
02 JavaScript sintaksė, stiliaus taisyklės

03 JavaScript kintamieji

04 JavaScript operatoriai ir duomenų tipai

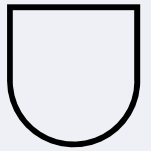
Bendra teorija: Front-End + Back-End (prisiminkime)





JavaScript galimybės

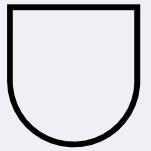
- Paprastas interneto svetaines (Websites);
- Sudėtingesnes interneto svetaines (Web & Server Apps);
- Mobiliąsias aplikacijas (Mobile Apps);
- Žaidimus (Browser games).



“JavaScript” patalpinimas/įtraukimas projekte

„JavaScript“ puslapyje patalpinamas panašiai, kaip ir CSS. Tačiau, kai CSS naudoja <link> tag’us išoriniam stiliui pritaikyti, o <style> tag’us – vidiniam stiliui pritaikyti HTML, „JavaScript“ reikia tik vieno HTML tag’o <script>. BŪDAI:

1. “Internal JavaScript” būdu;
2. **“External JavaScript” būdu (rekomenduojamas naudoti);**
3. “Inline JavaScript handlers” būdu.



JavaScript kintamieji (angl. *variables*)

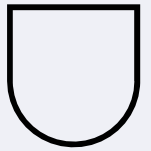
Kintamieji yra kaip “pavadinta duomenų saugykla”. Mes galime naudoti kintamuosius talpinti, prekes, lankytojams ir kitus duomenis.

Deklaruoti kintamuosius ir saugoti duomenis galime naudodami raktinius žodžius *var, let* arba *const*.

let – tai moderni kintamojo deklaracija.

const – yra tarsi let, bet kintamojo vertės pakeisti negalima.
var – tai senasis būdas inicijuoti kintamuosius, šiais laikais mes jo visiškai nenaudojame.

Kintamieji turėtų būti pavadinti taip, kad galėtume lengvai suprasti, kas yra jų viduje.



JavaScript sintaksė ir gramatika

JavaScript Variables (kintamosios vertės) var vs let vs const

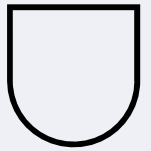
Pagrindinė problema su var:

```
var greeter = "hey hi";
var times = 4;

if (times > 3) {
    var greeter = "say Hello instead";
}

console.log(greeter) // "say Hello instead"
```

Jeigu greeter buvo naudojamas kitose kodo dalyse pamatysite jog output kodo bus visai kitoks nei tikėjotės, nes jis buvo iš naujo apibrėžtas if skiltyje. Tokiu atveju tai gali sukelti nemažai problemų jūsų kode, dėl to reikėtų naudoti let ir const.



JavaScript sintaksė ir gramatika

JavaScript Variables (kintamosios vertės) var vs let vs const

let yra var upgrade'as taip sakant ir išsprendžia pagrindinę var bėdą. let yra block scoped (bet kas kas yra {} viduje yra block). Tai reiškia, kad let galimas naudojimui tik bloko viduje. Šiuo atveju let hello yra block viduje ir output'o nedavė.

```
let greeting = "say Hi";
let times = 4;

if (times > 3) {
     let hello = "say Hello instead";
     console.log(hello);// "say Hello instead"
 }
console.log(hello) // hello is not defined
```

JavaScript sintaksė ir gramatika

JavaScript Variables (kintamosios vertės) var vs let vs const

Taip pat svarbu žinoti, kad let gali būti update'intas, bet ne iš naujo apibrėžtas.

```
let greeting = "say Hi";
greeting = "say Hello instead";
```

```
let greeting = "say Hi";
let greeting = "say Hello instead"; // error
```

Tačiau jei mes apibrėšime naudodami let tą patį variable skirtinguose scope, error'o nebus:

```
let greeting = "say Hi";
if (true) {
    let greeting = "say Hello instead";
    console.log(greeting); // "say Hello instead"
}
console.log(greeting); // "say Hi"
```

Error'o nėra, nes šie instances yra suprantami kaip skirtingi variables nes yra skirtinguose scopes. Ir kadangi tas pats variable negali būti apibrėžtas daugiau nei kartą tame pačiame scope, mes išvengiame pagrindinės problemos naudodami let kuri atsiranda su var.

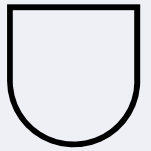
JavaScript sintaksė ir gramatika

JavaScript Variables (kintamosios vertės) var vs let vs const

const apibrėžti declarations išlaiko pastovią value. const kaip ir let yra block scoped (bet kas kas yra {} viduje yra block). Tačiau priešingai nei let, const negali būti nei apibrėžtas iš naujo, nei update'intas.

```
const greeting = "say Hi";
greeting = "say Hello instead";// error: Assignment to constant variable.
```

```
const greeting = "say Hi";
const greeting = "say Hello instead";// error
```



JavaScript sintaksė ir gramatika

„JavaScript“ vertės (values)

„JavaScript“ sintaksė apibrėžia dviejų tipų vertes: fiksuotasias ir kintamasias (fixed values and variable values).

Fiksuotos vertės yra vadinamos literaliais (**JavaScript Literals**). Kintamųjų vertės vadinamos kintamaisiais (**JavaScript Variables**).

JavaScript sintaksė ir gramatika

JavaScript Literals (fiksuotos vertės)

Svarbiausios fiksuotų verčių rašymo taisyklės yra:

- Skaičiai (numbers) rašomi su dešimtainiais ženklais arba be jų:
  - *10.50*
  - *1001*
- Tekstas (strings), rašomi dvigubomis, viengubomis arba atgalinėmis kabutėmis:
  - *"John Doe"*
  - *'John Doe‘*
  - *`John Doe`*

JavaScript sintaksė ir gramatika

JavaScript Variables (kintamosios vertės)

Programavime kintamieji naudojami duomenų reikšmėms saugoti.
„JavaScript“ kintamiesiems deklaruoti naudoja raktinį žodį var (ES6: *let* ir *const*).

Lygybės ženklas (=) naudojamas kintamiesiems priskirti reikšmes.

Šiame pavyzdyje x apibūdinamas kaip kintamasis. Tada x priskiriama (suteikiama) 6 vertė:

```
var x ;
x = 6;
```

JavaScript sintaksė ir gramatika

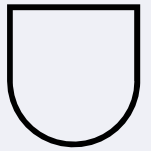
„JavaScript“ operatoriai

„JavaScript“ naudoja aritmetinius operatorius (+ - * /) reikšmėms apskaičiuoti:

Pvz.: console.log((5 + 6) * 10) // atsakymas bus 110

SVARBU: „JavaScript“ naudoja priskyrimo operatorių (=), kad priskirtų reikšmes kintamiesiems.

Pvz.: let x = 10



“JavaScript” operatoriai

Aritmetiniai operatoriai (Arithmetic operators)

Aritmetiniai operatoriai kaip operandus ima skaičių vertes (parašytų skaičiais ar kintamųjų pagrindu) ir grąžina vieną skaitinę reikšmę.

Standartiniai aritmetiniai operatoriai yra:

- sudėjimas (+);
- atimtis (-);
- daugyba (*);
- dalijimas (/).

JavaScript sintaksė ir gramatika

„JavaScript“ komentarai

Kodas po dvigubų brūkšnelių // arba tarp / * ir * / traktuojamas kaip komentaras.

Pvz.:

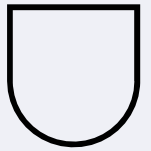
var x = 5; // – Ši dalis bus matoma, po brūkšnio bus komentaras kur galite rašyti bet ką

// var x = 6; – Visa kodo eilutė bus nematoma

JavaScript duomenų tipai (teorija)

Naujausias „ECMAScript“ standartas apibrėžia aštuonis duomenų tipus

1. Septyni primityvūs duomenų tipai:
    a. **Boolean**
    b. Null
    c. Undefined
    d. **Number**
    e. **BigInt**
    f. **String**
    g. Symbol
2. Ir Objektai (Object)



Primityvūs JavaScript duomenų tipai (teorija)

Boolean - reiškia loginį subjektą ir gali turėti dvi reikšmes: true arba false.

Šis tipas dažniausiai naudojamas norint išsaugoti „taip“ / „ne“ reikšmes.

Pvz.:
let nameFieldChecked = true;

let ageFieldChecked = false;

Boolean vertės taip pat gaunamos palyginus:
let isGreater = 4 > 1;

console.log(isGreater); // true.

Primityvūs JavaScript duomenų tipai (teorija)

Number.

Naujausias “ECMAScript” standartas pateikia du skaičių tipus: “Number” ir “BigInt”.

Number tipas žymi tiek sveikus, tiek skaičius su kableliu.

Yra daugybė operacijų, susijusių su Number, pvz.: *, / , +, - ir pan.

Be įprastų skaičių, yra ir vadinamųjų „specialiųjų skaitinių verčių“, kurios taip pat priklauso šiam duomenų tipui: “Infinity”, “-Infinity” ir “NaN”.

Pvz.:

let n = 123;

n = 12.345;

Primityvūs JavaScript duomenų tipai (teorija)

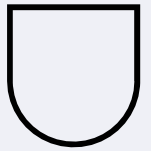
BigInt – Naujausias “ECMAScript” standartas pateikia du skaičių tipus: “Number” ir “BigInt”.

“JavaScript” programoje „Number“ tipas negali parodyti sveikųjų skaičių, didesnių nei 2^{53} (arba mažesnių nei -2^{53}, jei neigiama), tai yra techninis apribojimas, kurį sukelia jų vidinis atvaizdavimas.

Neseniai „BigInt“ tipas buvo pridėtas prie kalbos, kad būtų galima parodyti savavališko ilgio skaičius.

„BigInt“ yra sukurtas pridedant n prie sveikojo skaičiaus žodžio pabaigos:

const bigInt = 1234567890123456789012345678901234567890n



Primityvūs JavaScript duomenų tipai (teorija)

String - reprezentuoja tekstą ir apklijuota kabutėmis(“”, ‘’ arba ``).

“JavaScript”, turi 3 tipų kabutes:

Dvigubos kabutės: “Sveiki”.
Viengubos kabutės: ‘Sveiki’.

Atgalinės kabutės: `Sveiki`.

Pvz.:

let str = "Hello“;

let str2 = 'Single quotes are ok too';

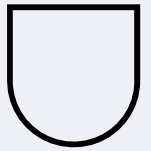
let phrase = `can embed another ${name}`;

Užduotis nr. 1

Sukurkite du kintamuosius x ir y ir priskirkite jiems reikšmes 50 ir 10.

Atlikite šiuos veiksmus, naudodami Sudėtinius paskyrimų operatorius, su sukurtais kintamaisiais:

- Sukuriamas naujas kintamasis “suma”, kuris yra lygus x kintamojo vertei sudėtai su y kintamuoju;
- Sukuriamas naujas kintamasis “skirtumas”, kuris yra lygus x kintamojo vertei skirtumui su y kintamuoju;
- Sukuriamas naujas kintamasis “daugyba”, kuris yra lygus x kintamojo vertei padaugintai iš y kintamuoju;
- Sukuriamas naujas kintamasis “dalyba”, kuris yra lygus x kintamojo vertei padalintai iš y kintamuojo.



Duomenų tipas – Numbers (paanalizuokime detaliau)

JavaScript kalboje yra dviejų tipų skaičiai (numbers):

- Įprasti skaičiai (Regular numbers), kurie naudojami dažniausiai;
- BigInt skaičiai (BigInt numbers), parodyti savavališko ilgio sveikus skaičius.

Nagrinėdami numbers duomenų tipą, didesnį dėmesį skirsime įprastiems skaičiams (Regular numbers).

Duomenų tipas – Numbers (paanalizuokime detaliau)

Įsivaizduokite, mums reikia parašyti 1 milijardą.

Akivaizdus būdas yra: *let milijardas = 1000000000*;

Naudodami JavaScript mes galime sutrumpinti skaičių, pridėdami raidę „e“ prie skaičiaus ir nurodydami nulių skaičių:

l*et milijardas = 1e9*; // 1 milijardas, kitaip: 1 ir 9 nuliai

console.log(7.3e9); // 7.3 milijardai (7,300,000,000)

Kitaip tariant, „e“ padaugina skaičių iš 1 su nurodytu nulių skaičiumi:

1e3 = 1 * 1000

1.23e6 = 1.23 * 1000 000

Duomenų tipas – Numbers (Regular numbers) (paanalizuokime detaliau)

Nurodykime ką nors labai mažo. Pvz.: 1 mikrosekundė (milijoninė sekundės dalis):

let ms = 0.000001;

Kaip ir anksčiau, „e“ naudojimas gali padėti. Jei norėtume vengti rašyti nulius, galėtume parašyti tą patį:

let ms = 1e-6; // šeši nuliai į kairę nuo 1

Kitaip tariant, neigiamas skaičius po „e“ reiškia 1 padalijimą iš nurodytų nulių skaičiumi:

// 1 padalinta 1 su 3 nuliais

1e-3 = 1 / 1000 (=0.001)