## **Kintamieji - ​​Variables**

Dažniausiai, veikiančiai programai reikia sekti kaip keičiasi atitinkamos reikšmės, jeigu yra atliekamos skirtingos operacijos, arba iššaukiami atitinkami veiksmai.

Tam tikslui yra sukurti kintamieji, kurie savyje laiko simbolinę reikšmę.   
Ji gali būti pakeista vėlesniuose žingsniuose, kai programa yra leidžiama.

| **let a= 20;** |
| --- |

Kaip matome pavyzdyje, mes **a** kintamajam priskyrėme skaičiaus reikšmę kuri yra lygi **dvidešimt**.

Vėliau šią reikšmę galime pakeisti.

| let kaina = 20; kaina = kaina + (kaina \* 0.21) console.log(kaina) *// nauja “kainos” reikšmė yra 24.2* |
| --- |

### **Kintamųjų kūrimas su skirtingais raktažodžiais**

Javascript kintamieji yra kuriami su trimis raktažodžiais (keywords) t.y. ***var, let, const.***

| ***var name; let name; const name;*** |
| --- |

***var* raktažodis** leidžia deklaruoti naują kintamąjį, kuris bus apribotas funkcijos arba globaliame kontekste. Su šiuo raktažodžiu sukurti kintamieji, vėliau gali keisti reikšmes, ir prisiskirti naujas. Jei kintamasis nebuvo sukurtas funkcijos viduje, jis patampa globalus, ir jei tame pačiame kontekste bus sukurtas lygiai toks pat kintamasis su raktažodžiu var, senasis kintamasis bus ištrintas ir ant viršaus užrašytas naujas.

***let* raktažodis** leidžia deklaruoti nauja kintamaji, kuris bus apribotas tik kodo bloke, kuriame jis buvo aprašytas. Tokiam kintamajam mes galime keisti reikšmę vėliau, tačiau negalime redaklaruoti, kaip tai įmanoma su **var**.

***const*** **raktažodis** leidžia deklaruoti nauja kintamaji, kuris bus apribotas tik kodo bloke,

kuriame jis buvo aprašytas. Tokio kintamojo reikšmė negali būtų keičiama po to yra deklaruojamas (nors tai nėra visiškai tiesa, daugiau informacijos nagrinėsime prie masyvų ir objektų).

Pagrindiniai skirtumai tarp skirtingai uždeklaruotų kintamųjų, kad **let** ir **const** yra bloko scope, o **var** yra function scope. (Scope yra apibrėžta kodo vieta iš kurios galime pasiekti kintamojo reikšmę, jei kintamasis bandomas pasiekti iš vietos kuri neturi teisės jį pasiekti, Javascript mes klaidą.)

Naudojimas ***var*** raktažodis kuriant kintamuosius yra laikoma kaip blogoji praktika ir rekomenduojama nedaryti, kintamus reiktų kurti su ***const*** jei reikšmė nekis arba ***let*** jei reikšmė keičiasi.

### **Kintamųjų vadinimas**

Kintamųjų pavadinimuose gali būti tik raidės A-Z ir “\_” bei “$”, ir skaitmenys 0-9 (tačiau negali prasidėti skaičiumi) visa kita yra draudžiama.

1. **Camel case**

| **const submitBtnColor = "red";** |
| --- |

1. **Upper camel case**

| **const SubmitBtnColor = "red";** |
| --- |

1. **Snake case**

| **const submit\_btnColor = "red";** |
| --- |

1. **Kebbab case** (Javascript kalboje nėra leidžiama)

| **const submit-btn-color = "red";** |
| --- |

Kuriant kintamuosiuose visada gerai apgalvokite jų pavadinimus, kad nebūtų per ilgi, aiškiai apibūdintų jo reikšmę ir tikslą.

### **Kintamųjų reikšmių tipai**

Kintamiesiems, javascript kalboje galima priskirti tik atitinkamus kintamųjų tipus. Jie yra:

| String Number Boolean null undefined BigInt Symbol Object  Function  Array  Object literals  Map |
| --- |

**String** - atstovauja teksta Javascript kalboje, visas tekstas kuris parašytas kabutėse (viengubose, dvigubose, arba atgalinėmis) yra laikomas string duomenų tipas.

| 'tekstas su viengubomis kabutėmis' "tekstas su dvigubomis kabutėmis" `tekstas su atgalinėmis kabutėmis` |
| --- |

**Number** - atstovauja skaičius. Rašant skaičius, kabučių NEREIKIA.

| 1 20 1000 999999999999999 |
| --- |

**Boolean** tipas turi dvi reikšmes, true arba false arba Kitais žodžiais tariant taip arba ne.

Šis duomenų tipas yra naudojamas sąlygų išraiškose. Javascript kalba išraiškas gali automatiškai paversti į boolean tipa.

Pvz. 2 === 2 yra lygu true arba 2 > 10 yra lygu false.

**Null** yra duomenų tipas yra laikomas nulinis, niekinis, neturintis reikšmės duomenų tipas.

Kuomet norime pasakyti, kad kintamasis neturi reikšmės, mes jam priskiriama null reikšmę.

**Undefined** - reikšmė yra automatiškai sugeneruojama arba ją galima ir priskirti, kuomet buvo sukurtas kintamasis, tačiau jam reikšmė dar nepriskirta.

Pvz. const a; console.log(a) *// undefined*

sugalvoti Kadangi Javascript kalboje kintamųjų be reikšmiių negali būti, todėl jei programuotojas nepriskyrė reikšmės jam automatiškai yra priskiriama undefined reikšmė.

**BigInt** tas pats, kas Number tik skirtas labai dideliems skaičiams.

**Symbol** yra specifinė reikšmė, kuri nėra kintamasis, bet turi kintamojo savybių. Jis gali būti deklaruotas viena karta, ir dažniausiai naudojams su ES6 modules import. Šiame kurse mes apie juos nesimokinsime.

**Object** yra didžiulis ir pagrindinis duomenų tipas Javascript kalboje. Jis susideda iš dar kitų potipių kaip funkcija, masyvas (array), tiesioginis objektas, žemėlapis.

Tipai buvo sugalvoti tam, kad su jais galima būtų atlikti skirtingas operacijas, arba tie patys operatoriai atliktų skirtingus veiksmus.

Pvz: '1' + '1' = '11' arba 1 + 1 = 2

Daugiau apie kiekvien1 duomenų tipą kalbėsime kurso eigoje.