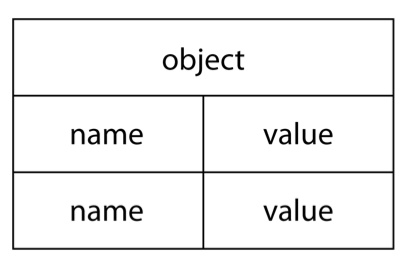
# **Objects**

Įvadas į objektus

Objektas yra vienas iš duomenų tipų Javascript kalboje. Javascript kalboje objektai yra dinaminiai, o tai reiškia, kad jie gali būti pakeisti bet kuriuo momentu kuomet kodas yra paleidžiamas. Kitos klasėmis grindžiamomis kalbomis objektai yra užrakinami kai jie jau sukuriami, tačiau Javascript’e tai negalioja.

Didelė dalis Javascript programavimo yra darbas su objektais, todėl suprasti kaip objektai kuriami, kaip jie veikia yra raktas suprasti Javascript kalbą.

Pvz: Objekto struktūra

****

Objektas susideda iš atributų rinkinių ir tiem atributams priskirtų reikšmių. Objektas gali turėti tik viena to pavadinimo atributą, tačiau gali turėti vienodas reikšmes skirtingiems atributams. Objekto reikšmė gali būti bet kuris kitas Javascript duomenų tipas (string, number, null, undefined, boolean, array, function, object and etc.)

Programavimo kalbose, objektai naudojasi ta pačia logi kaip ir objektai mūsų kasdieniniam gyvenime.

| **Object - car** | **Properties (atributai)** | **Methods (Metodai)** |
| --- | --- | --- |
|  | car.name = Fiat  car.model = 500  car.weight = 850kg  car.color = white | car.start()  car.drive()  car.brake()  car.stop() |

## **Sukurti Javascript objekta**

**Object constructors**

Pvz: šis kodas sukuria generic objektą, naudojant konstruktoriaus funkcija

| var object = new Object(); |
| --- |

Šitas objektas buvo sukurtas pasinaudojus konstruktoriaus (constructor) funkcija.

Konstruktorius yra tiesiog funkcija, kuri naudojama kartu su new žodeliu. Javascript kalboje, bet kuri funkcija gali būti naudojama kaip konstruktorius. Pagal konvencija visos konstruktoriaus funkcijos yra pradedamos rašyti iš didžiosios raidės.

“Object” yra jau pačioje kalboje apibrėžta funkcija/konstruktorius, tačiau konstruktoriaus funkcija mes galime pasirašyti ir patys.

**Object literals**

Pvz: šis kodas sukuria objektą naudojant object literal (literala).

| const person = {  firstName: "John",  lastName: "Doe",  age: 50,  eyeColor: "blue" }; |
| --- |

Javascript kalba leidžia kurti objekta ir tiesiogiai aprašyti objekto atributus vien panaudojus riestinius skliaustus. Priskirdami kintamajam **{}** tuščius skliaustelius, sukuriame tuščia objektą. Tuo pačius iš karto galime aprašyti atributus ar metodus. Toks būdas naudojamas daugiau kaip duomenų saugojimo tipas kuomet mes norime apsirašyti tik viena tokį objektą ir jo daugiau nereplikuoti.

Iš principo funkcijų kūrimas panaudojant function myFunction() {} yra funkcijos kūrimas literaliu būdu. (function literal)

| function reflect(value) {   return value; } *// tas pats kas* var reflect = new Function("value", "return value;"); |
| --- |

Objektų kūrimas su konstruktoriumi labiau naudojamas, kuomet mes norime apsirašyti objekto kūrimo brėžinį ir vėliau dinamiškai tokius aprašytus objektus generuoti.

Ir vienu ir kitu būdu sukurti objektai yra tokie patys, ir turi visas objekto savybes. Front end programavima, labiau paplitęs yra “object literal” būdas, todėl kad jis leidžia sugrupuoti atributus ir juos perduoti kaip viena subjektą (entity).

## **Priskirti ir pasiekti objekto atributus ir metodus**

| const person = {  firstName: "John",  lastName : "Doe",  id : 5566,  fullName : function() {  return this.firstName + " " + this.lastName;  } }; |
| --- |

Kuriant objektą kaip object literal, mes viska galime aprašyti iš karto. Tačiau visa tai galime padaryti ir vėliau per taško arba laužtinių skliaustų notacija.

| const person = {};  person.firstName = "John"; person.lastName = "Doe"; person.id = 5566; person.fullName = function() {  return this.firstName + " " + this.lastName; }; |
| --- |

Norėdami pasiekti ir panaudoti atributo reikšmę, tiesiog parašme objekto pavadinimą ir su taško notacija atributo pavadinimą. Taip pat galime nurodyti ir laužtinius skliaustus, tačiau atributo pavadinimą rašome su kabutėmis, kaip string.

| person.firstName *// "John"* person["lastName"] *// "Doe"* |
| --- |

**Patikrinti ar objektas turi atributus**

| *// Jei neesame tikri, kad objektas turi atitinkama atributą, visada galime tai pasitikrinti su sąlyga* if (person1.age) {  *// darome kažką su age atributu* } |
| --- |

| *//Pasitikrinam ar objektas turi metodą kurį norime iššaukti* person.walk ? person.walk() : '';  person.walk && person.walk(); |
| --- |

Jei kreipiamės į objektą atributą, bet jo reikšmė nėra apibrėžta, toks atributas iškarto gaus reikšmę kaip undefined.

| console.log(person.randomAttribute); *// undefined* |
| --- |

## **Variable types by reference and by value**

Objektas yra “reference” tipo duomenų struktūra, kas reiškia, kuomet jis yra sukuriamas jis nėra tiesiogiai priskiriamas kintamajam ir kintamasis tiesiogiai nesaugo objekto reikšmės. Vietoj to jis saugo nuorodą į operatyvios atminties lokacija kur tas objektas buvo sukurtas.

*Tai yra pagrindinis skirtumas tarp objektų ir primityvių duomenų tipų, javascript kalboje.*

Kai priskiriamas objektas kintamajam, jam ištikrųjų yra sukuriama nuorodą į atminties adresą kur tas objektas saugomas, kas reiškia, jei tą kintamąjį priskirtumėte kitam kintamajam, abu kintamieji turėtų nuorodas į ta patį objektą ir nebūtų sukuriama to objekto kopija, kaip tai vyksta su primityviais tipais.

| ***// Su primityviais tipais, b sukuria nauja reikšmės kopija ir saugo nauja reikšmę.* let a = "some text"; let b = a; a = "another text" console.log(a); *// another text* console.log(b); *// "some text"*   *// Su objektais y nesukūrė naujo objekto, o priskyrimo momentu prisiskyrė objekto // reference (nuoroda) į objektą, todėl pakeitus originalų objektą keičiasi visi // kintamieji, nes jie visi rodo į tą patį pakeista objektą.*   let x = {  name: "John" } let y = x; x.name = "Peter";  console.log(x.name); *// Peter* console.log(y.name); *// Peter*** |
| --- |

## **Kalboje jau esantys built-in objektai arba dar vadinami *Global Objects.***

• Array - An ordered list of numerically indexed values

• Date - A date and time

• Error - A runtime error (there are also several more specific • Function - A function

• Object - A generic object

• RegExp - A regular expression

• String - A text

• Math - A tool working with mathematical operations

• Map - A one of data type

Daugiau skaityti: ​​<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects>