# Kodo blokai, Sąlygos.

Code blocks, Conditions

## **Kodo blokas - Code block**

Kodo blokas tai yra grupė programavimo sakinių. Bloko ribos yra pažymimos riestiniais skliaustais "{}". Blokas prasideda atsidarančiu riestiniu skliaustu **"{"** ir baigiasi užsidarančiu "}". Nors kiekvienas programavimo sakinys bloko viduje baigiamas kabliataškiu, pats blokas – ne, todėl tokią struktūrą galima vadinti vienu sudėtiniu sakiniu (statement). Jeigu yra poreikis kodo blokui galima suteikti atskirą pavadinimą. Kodo blokas visada eina kartu su funkcija (function), sąlygomis (conditions) ir cikliais (loops), tačiau galima naudoti ir kaip atskirą programavimo elementą.

Pvz: Kodo blokas

| {  *// Kodo blokas* } |
| --- |

Pvz: Kodo blokas su pavadinimu

| Pavadinimas: {  *// Kodo blokas* } |
| --- |

Pvz: Kodo blokas su kintamaisiai pavyzdys

| var x = 1;  let y = 1;  {  var x = 2;  let y = 2  }  console.log(x); *// logs 2, todėl, kad var yra function scoped kintamasis*  console.log(y); *// logs 1, todėl, kad let yra block scoped kintamasis* |
| --- |

## **Sąlygos - Conditions**

Sąlygos yra programavimo kalbos mechanizmas skirtas nustatyti ar paleisti po sąlyga einantį kodo sakinį, bloka. Sąlygos sakinio, prasidedančio baziniu žodžiu **"if"**, bendrasis pavidalas yra toks:

| if (sąlygos\_išraiška) { kodo blokas }; |
| --- |

Jis vykdomas taip: visų pirma apskaičiuojama sąlygos\_išraiška. Sąlygos išraiškos reikšmė visada bus boolean. Jei jos reikšmė yra true, tuomet kodo blokas yra vykdomas. Jei jos reikšmė yra false, tuomet nurodytas kodo blokas nevykdomas ir programa tęsiama sakiniu, einančiu po kodo bloko.

Pvz: Sąlyga su **if** sakiniu

| const x = 10;  if (x === 10) {  *// Sąlyga įvykdyta, paleidžiami kodo sakiniai esantys šiame kodo bloke.* } |
| --- |

Taip pat yra galimybė nurodyti, ka reikėtu daryti, kai salygos\_išraiškos rezultatas yra false. Tam gali būti naudojamas bazinis žodis else, kuriuo išplečiamas if sakinys:

Pvz: Sąlyga su if else sakiniu

| const x = 20;   if (x === 10) {  *// Šiame bloke kodo sakiniai nebus paleidžiami, nes sąlyga neišpildyta*  } else {  *// Sąlygos reikšmė yra false, paleidžiami kodo sakiniai šiame bloke*  } |
| --- |

Po else gali būti rašomas naujas if sakinys. Kitas pavyzdys rodo, kaip bus elgiamasi, kai x yra daugiau, lygus ir mažiau už 0:

Pvz: Sąlyga su else if sakiniu

| if (x > 0) {  console.log("X daugiau už 0"); } else if (x < 0) {  console.log("X mažiau už 0"); } else {  console.log("X yra lygus 0"); } |
| --- |

Jeigu su sąlyga yra naudojamas tik vienas sakinys, kodo bloko su riestiniais skliaustais galima ir ne naudoti.

Pvz: Sąlyga be kodo bloko {}

| *// Su if sakiniu* if (x === 10) console.log("X yra lygu 10");  *// Su if else sakiniu* if (x === 10) console.log("X yra lygu 10"); else console.log("X nėra lygu 10");  *// Su else if sakiniu* if (x > 0) console.log("X daugiau už 0"); else if (x > 0) console.log("X mažiau už 0"); else console.log("X yra lygus 0"); |
| --- |

Sąlygos išraiškoje be kodo bloko, galimas tik vienas programavimo sakinys kuris baigiamas kabliataškiu. Jei parašysite kelis kodo sakinius, Javascript meta klaidą.

Jei norime supaprastinti sąlyga, kad rašytumėtei mažiau kodo, galima naudoti “?” ternary operatoriu sąlygai išreikšti.

Pvz: Sąlyga su **? ternary**  operatoriumi

| x === 10 ? console.log("X yra lygu 10"); : console.log("X nėra lygu 10"); |
| --- |

## Sąlygos su palyginimo ir logikos operatoriais - Conditions, comparison and logical operators

Sąlygas galime jungti kartu su loginiais operatoriais. Kuriuos sudedant kartu gaunama sąlygos išvestį.

Pvz: Sąlyga kartu su loginiais operatoriais

| if ((hours > 6) && (hours < 11)) {  console.log("Labas rytas");  } else if ((hours > 11) && (hours < 18)) {  console.log("Laba diena");  } else if ((hours > 18) && (hours < 21)) {  console.log("Labas vakaras");  } else {  console.log("Labanaktis");  } |
| --- |

Pvz: Loginių operatorių veikimas sąlygoje

| true && true; *// reikšmė true sąlyga vykdoma* true && false; *// reikšmė false sąlyga nevykdoma* false && true; *// reikšmė false sąlyga nevykdoma* false && false; *// reikšmė false sąlyga nevykdoma*  true || false; *// reikšmė true sąlyga vykdoma* false || true; *// reikšmė true sąlyga vykdoma* false || false; *// reikšmė false sąlyga nevykdoma*  !true; *// reikšmė false sąlyga nevykdoma* !false; *// reikšmė true true sąlyga vykdoma*  !!true; *// reikšmė true true sąlyga vykdoma*  !!false; *// reikšmė false sąlyga nevykdoma* |
| --- |

Loginius operatorius galima taikyti ant visų reikšmių tipo (string, number ir t.t.) sąlygoje jie bus išversti į boolean reikšmę.

### Truthy reikšmės

Javascript naudoja reikšmių tipų konvertavimą į Boolean sąlygose, cikluose ar kituose boolean kontekstuose. (x === 10) toks kodas, bus paverstas true arba false.

**Truthy** reikšmė yra laikoma kaip true**,** kuomet yra naudojama sąlygoje ar boolean kontekste. Visos įmanomos reikšmės Javascript yra laikomos kaip **truthy reikšmės,** jei jos nėra **falsy** reikšmių sąraše. Tai yra teksto reikšmės (išskyrus tuščia), skaičiai (išskyrus 0), objektai, masyvai, funkcijos. (string, number, object, function, array).

Pvz: Truthy reikšmės

| *//Truthy reiškia true* if (true) *// lygu true* if ({}) *// lygu true* if ([]) *// lygu true* if (42) *// lygu true* if ("0") *// lygu true* if ("false") *// lygu true* if (new Date()) *// lygu true* if (-42) *// lygu true* if (12n) *// lygu true* if (3.14) *// lygu true* if (-3.14) *// lygu true* if (Infinity) *// lygu true* if (-Infinity) *// lygu true* |
| --- |

### Falsy reikšmės

Falsy reikšmės yra laikomos false, kuomet naudojamos sąlygoje arba boolean kontekste.

Pvz: Truthy reikšmės

| *//Falsy reiškia false* if (false) if (null) if (undefined) if (0) if (-0) if (0n) if (NaN) if ("") |
| --- |

### Switch salygos sakinys.

Switch sąlygos struktūra yra kažkiek savita. Priklausomai nuo kintamojo reikšmės, vykdoma viena iš sakinių grupių, kuri gali būti norimas kiekis. Tai kažkiek panašu į if ir else if sakinius. Switch sakinys įvertina išraiškos vertę su ta išraiška kuri yra susieta su case sąlyga ir paleidžia kodo sakinius susietais su tuo case.

Pvz: Switch sakinys

| switch(sąlygos\_išraiška) {  case x:  case c:  case z:  *// jei atitiko viena iš sąlygų (case), tuomet vykdomas šis kodo blokas*  break; *// sustabdoma šioje vietoje, kad netikrintų kitų sąlygų*  case y:  *// vykdomas kodo blokas*  break;  default:  *// jei neatitiko nei vienos sąlygo, tuomet vykdomas šis kodo blokas* } |
| --- |

Switch sąlyga vykdomas taip: visų pirma apskaičiuojama išraiška ir tikrinama, ar gauta reikšmė lygi x. Jeigu taip, vykdoma grupė 1 iki sutinkamas break sakinys. Sutikus jį, peršokama į switch sakinio galą.

Jei išraiškos reikšmė nėra lygi x, tuomet tikrinama, ar ji nėra lygi c. Jei lygu, tai vykdoma grupė 1 iki break ir peršokama i switch sakinio galą. Jei išraiškos reikšmė nėra lygi x,c,z. Tuomet tikrinama išraiška y, ir jei ji atitinka, vykdoma grupė 2 iki break ir vėl peršokama 5 switch sakinio galą.

Galų gale, jei išraiškos reikšmė nėra lygi nė vienai iš nurodytų konstantų (case konstantų gali būti tiek, kiek mums reikia), vykdomi sakiniai, esantys po žymės default, jei ji yra. default dalis switch sakinyje nebūtina.

Kaip matome pavyzdyje, vienam kodo blokui, galime apjungti daugiau negu viena sąlyga.

Switch sąlygomis galime ne tik vykdyti kodo bloką su sakiniais bet ir gražinti atitinkama apskaičiuotą reikšmę, jei pavyzdžiui norime ją grąžinti funkcijoje. Tuomet break žodžio nereikia, nes return gražinimas užbaigia funkcijos vykdyma kuris jis yra panaudojamas.

Pvz: Switch sakinys su grąžinama reikšme

| *// Funkcija grąžinanti savaitės dienos pavadinimą* function getWeekDay() {  const date = new Date();  const d = date.getDay();   switch (d) {  case 1:  return "Pirmadienis";  case 2:  return "Antradienis";  case 3:  return "Trečiadienis";  case 4:  return "Ketvirtadienis";  case 5:  return "Penktadienis";  case 6:  return "Šeštadienis";  case 7:  return "Sekmadienis";  } }  const day = getWeekDay(); console.log(day); *// Dienos pavadinimas išspausdinamas į ekrana* |
| --- |