# **Ciklai**

Loop

## Ciklas yra kodo blokas, kuris gali paleisti kodą atitinkamą kartų skaičių. Dažniausiai jie yra naudojami, kada dirbama su masyvais.

Ciklų rūšys:

for - leis kodą atitinkama kartų skaičių, nurodytą sąlygoje.

for/in - leis kodą per objekto atributus.

for/of - leis pro iteruojamo objekto reikšmes.

while - leis koda pro kodo bloką, iki kol nurodyta sąlyga bus įvykdyta.

do/while - leis koda pro kodo bloką, iki kol nurodyta sąlyga bus įvykdyta.

## **For** ciklas (loop)

For ciklo sintaksė

| for (statement 1; statement 2; statement 3) {  *// vykdomas kodas* } |
| --- |

1. Statement 1 leidžiamas viena karta, prieš paleidžiant ciklo kodo bloką.
2. Statement 2 aprašo sąlyga, kad būtų leidžiamas kodo blokas
3. Statement 3 yra leidžiamas kiekviena karta, kada kodo blokas jau būna įvykdytas.

| const cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];  let text = ""; for (let i = 0; i < cars.length; i++) {  text += cars[i] + "<br>"; } |
| --- |

Pirmas statement1 sukuria kintamuosius i, prieš ciklui prasidedant. Statement2 nurodo sąlyga, kad ciklas pradėtų vykdyti kodo komandas, statement3 pakels i reikšmę, kada bloko kodas bus įvykdytas.

**Statement 1** - paprastai skirtas inicijuoti kintamuosius, kurie dalyvaus ciklo sąlygoje. Šitas statement nėra būtinas, bet paprastai programuotojai jį nurodo. Galima sukurti daug kintamųjų statement 1 dalyje.

**Statement 2** - paprastai šis sakinys yra skirtas nustatyti sąlyga kurią įvykdžius pasileidžia ciklas ir kodo bloko vykdymas. Šitas statement irgi nėra būtinas, tačiau tuomet reikia nurodyti sąlyga viduje kodo bloko ir ciklą sustabdyti su baziniu žodžiu break. Kitu atveju ciklas niekada nesibaigs ir anksčiau ar vėliau nulauš jūsų interneto naršyklę.

**Statement 3** = paprastai pakels reikšmę pradinio kintamojo. Šis statement gali daryti ką norite, gali pakelti per viena, gali pakelti reikšmę per du, gali atimti reikšmę ir t.t. Šitas statement irgi nėra privalomas, tačiau kintamojo modifikacija tuomet reikia atlikti kodo bloko viduje.

Pvz: For Ciklas

| let i = 0; let len = cars.length; let text = ""; for (; i < len; ) {  text += cars[i] + "<br>";  i++; } |
| --- |

### Naudojant var kintamasis for cikle.

| var i = 5;  for (var i = 0; i < 10; i++) {  *// kodas* }  *// Čia i yra 10* |
| --- |

### Naudojant let kintamasis for cikle.

| let i = 5;  for (let i = 0; i < 10; i++) {  *// kodas*  }  *// Čia i yra 5* |
| --- |

Kaip matome, kadangi var kintamojo tipas nėra block scope, jis yra užrašomas ant viršaus, kada ciklas yra iteruojamas. Kadangi let ura block scope kintamojo tipas, cikle esantis let kintamasis jau yra kitame scope lygmenyje, todėl nėra paveikta ciklo vykdymo.

## **For In** ciklas (loop)

Šio tipo ciklas, iteruoja pro objekto atributus.

| for (key in object) {  *// code block to be executed* } |
| --- |

Pvz:

| const asmuo = {vardas: "John", pavarde: "Doe", amzius: 25};  let text = ""; for (let x in person) {  text += asmuo[x]; } |
| --- |

For in ciklas iteruoja per asmuo objekta, kiekvienoje iteracijoje x yra asmens objekto vis kitas atributas. Cikloje viduke atributas yra naudojamas pasiekti objekto atributo reikšmę (asmuo[x])

For In ciklas gali būti naudojamas ir iteruojant per masyva (array).

| for (variable in array) {  code } |
| --- |

| const numbers = [45, 4, 9, 16, 25];  let txt = ""; for (let x in numbers) {  txt += numbers[x]; } |
| --- |

Tačiau toks naudoti šita ciklą masyvui yra nerekomenduojama, nes jis kiekviena indekso reikšmę verčia į teksta ir tokia operacija yra labai lėta. Tam geriau naudoti for ciklą.

## **For Of** ciklas (loop) (ES6)

Javascript for of loop statement operuoja per reikšmes iteruojamų objektų. Jis leidžia iteruoti per iteruojamus duomenis jei interuojamas objektas yra Array (masyvas), String (tekstas), Map, NodeList ir kitus.

| for (let variable of iterable) {  *// code block to be executed* } |
| --- |

**variable** - kiekvienai iteracijai viariable yra iteruojamo atributo reikšmė.

**iterable** - yra objektas kuris turi iteruojamus atributus.

Pvz: iteruojamas masyvas - array

| const cars = ["BMW", "Volvo", "Mini"];  let text = ""; for (let x of cars) {  text += x; } |
| --- |

Pvz: iteruojamas tekstas - string

| let language = "JavaScript";  let text = ""; for (let x of language) { text += x; } |
| --- |

## **While** ciklas (loop)

While ciklas yra iteruojamas per kodo bloką, iki kol sąlyga yra lygi true.

| while (sąlyga) {  *// code block to be executed* } |
| --- |

Pvz:

| while (i < 10) {  text += "The number is " + i;  i++; } |
| --- |

## **Do While** ciklas (loop)

do while ciklas yra kita variacija while ciklo. Šitas ciklas ivykdys kodo bloka įvykdima viena karta, prieš tikrinant sąlyga. Jei sąlyga lygi true, tuomet vykdys kodo vykdima dar karta ir iki tol, kol sąlyga bus false.

| do {  *// kodo blokas* } while (sąlyga); |
| --- |

| do {  text += "The number is " + i;  i++; } while (i < 10); |
| --- |

**Break** reikšmė cikle

break; reikšmė kuomet atsiranda cikloje kodo bloko viduje iš karto užbaigia ciklą toje vietoje kur ji buvo parašyta.

Pvz:

| for (let i = 0; i < 10; i++) {  if (i === 3) { break; }  text += "The number is " + i + "<br>"; } |
| --- |

## Pavyzdyje viršuje jei sąlyga yra įvykdoma, kada i lygi  3, tuomet ciklas yra iš karto užbaigiamas.

## **Continue** reikšmė cikle

continue; reikšmė cikle užbaigia viena ciklo iteracija, jei pats ciklas dar nėra iki galo užbaigtas, tuomet ciklas yra leidžiamas toliau.

Pvz:

| for (let i = 0; i < 10; i++) {  if (i === 3) { continue; }  text += "The number is " + i + "<br>"; } |
| --- |