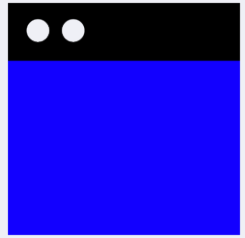
Code Academy



Gytis Juozenas

JavaScript switch, ternary operator and loops

2022

JavaScript programavimo kalba

Šiandien išmoksite

01 Developer‘s Tools Console

02 Ternary Operator

03 for Loops

04 Switch

“JavaScript” operatoriai

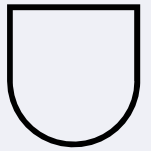
Sąlyginis operatorius (Conditional (ternary) operator)

Sąlyginis operatorius (Conditional (ternary) operator) yra vienintelis JavaScript operatorius, paimantis tris operandus.

Operatorius gali turėti vieną iš dviejų verčių, pagrįstų sąlyga.
Sintaksė yra:

condition ? exprIfTrue : exprIfFalse

Jei sąlyga (condition) teisinga, operatorius turi vertę val1. Kitu atveju ji turi val2 vertę.



“JavaScript” operatoriai

Sąlyginis operatorius (Conditional (ternary) operator)

Sąlyginį operatorių galite naudoti bet kurioje standartinio operatoriaus vietoje.

Ternary operatorius dažnai naudojamas kaip trumpinys if statement:

```
var age = 26;
var beverage = (age >= 21) ? "Beer" : "Juice";
console.log(beverage); // "Beer"
```

Užduotis nr. 1

Paverskite žemiau pavaizduotą if else į ternary operator:

```
let a = true;

let b;

if (a === true) {
  b = true;
} else {
  b = false;
}
```

```
var age = 26;
var beverage = (age >= 21) ? "Beer" : "Juice";
console.log(beverage); // "Beer"
```

Užduotis nr. 2

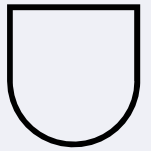
Paverskite žemiau pavaizduotą if else į ternary operator:

```
let num1 = 1;

let num2 = 2;


if (num1 < num2)
  { console.log(“True
  “)
} else
 { console.log(“False
 “)
}
```

```
var age = 26;
var beverage = (age >= 21) ? "Beer" : "Juice";
console.log(beverage); // "Beer"
```



JavaScript ciklai (Loops) (teorija)

Su ciklais (loops) galime automatizuoti ir pakartoti kodo bloką, kad jis veiktų kiek norime kartų, net ir neribotą laiką.
JavaScript suteikia daugybę būdų naudoti ciklus (loops).

Aptarsime tris ciklus (loops):

- while
- do...while
- **for**

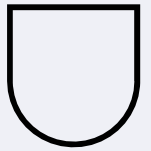
JavaScript ciklai (Loops) (teorija)

for (loop) ciklas
For ciklas yra sudėtingesnis, tačiau jis yra dažniausiai naudojamas.

Sintaksė:

```
for (begin; condition; step) {
  // ... loop body ...
}
```

```
for ([initialExpression]; [conditionExpression]; [incrementExpression]){
  // statement
}
```

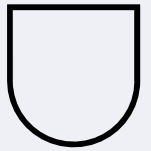


JavaScript ciklai (Loops) (teorija)

for (loop) ciklas

Pvz.:

```
for (let i = 0; i < 3; i++) { // shows 0, then 1, then 2
  console.log(i);
}
```

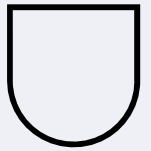


JavaScript ciklai (Loops) (teorija)

for (loop) ciklas

Kodo sudedamosios dalys:

part		
begin	`i = 0`	Executes once upon entering the loop.
condition	`i < 3`	Checked before every loop iteration. If false, the loop stops.
body	`alert(i)`	Runs again and again while the condition is truthy.
step	`i++`	Executes after the body on each iteration.



JavaScript ciklai (Loops) (teorija)

for (loop) ciklas

```
for (let elephant = 1; elephant < 10; elephant+=2)
  { console.log('elephant is ' + elephant);
}
```

output:

elephant is 1

elephant is 3

elephant is 5

elephant is 7

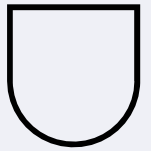
elephant is 9

Užduotis nr. 3

Parašykite for loop‘ą kuriame bus sukurtas kažkoks variable kuris outputins consolėje skaičius 0 4 8 12 16.

Pvz.:

```
for (let i = 0; i < 3; i++) { // shows 0, then 1, then 2
  console.log(i);
}
```



JavaScript sąlygos sakiniai (conditionals)

The "switch" statement

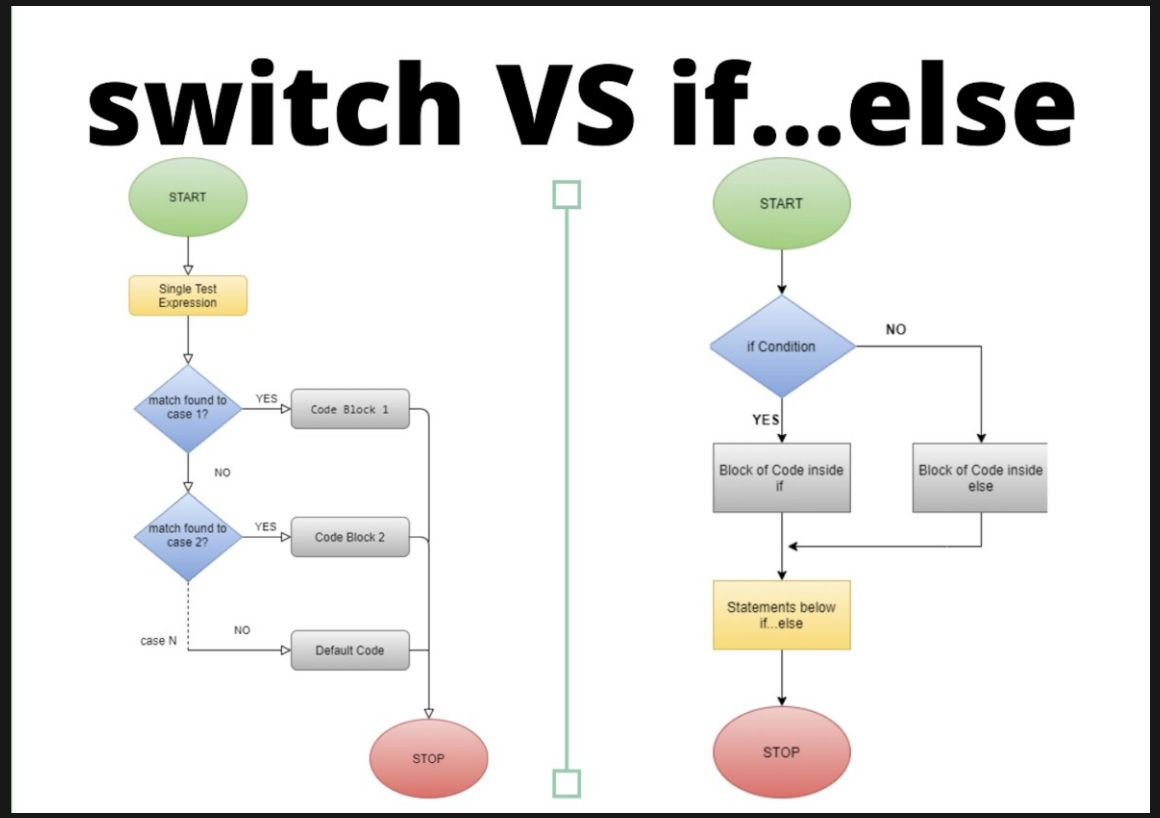
Kaip alternatyvą tikrinimui su *if* (*if...else* arba *else if*) galima naudoti *switch statement.*

Sintaksė:

```
switch(x) {
  case 'value1':  // if (x === 'value1')
    ...
    [break]

  case 'value2':  // if (x === 'value2')
    ...
    [break]

  default:
    ...
    [break]
}
```



JavaScript sąlygos sakiniai (conditionals)

```
let n = 20;
if (n == 10){
  console.log("n = 10");
} else if (n == 90) {
  console.log("n = 90");
} else if (n == -3) {
  console.log("n = -3");
} else {
  console.log("none of the above");
}
```

```
let n = 20;
switch (n) {
  case 10:
    console.log("n = 10");
  case 90:
    console.log("n = 90");
  case -3:
    console.log("n = -3");
  default:
    console.log("none of the above");
}
```

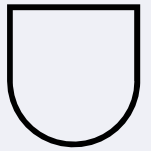
Užduotis nr. 4

Naudodamiesi switch parašykite kodą kuris tikrintų ar a yra lygus 2 + 2. Susikurkite pradžiai let a = 2 + 2;

Pabandykite du variantus su break ir be break, t.y. parašykite bent 3 skirtingus case ir default, o case turėtų būti per mažas, lygus 4, per didelis ir default kad nei vienas iš tų variantų. Pasižiūrėkite kuom skiriasi jūsų console.log kai naudojate break ir nenaudojate break kiekvienoje case situacijoje.

```
let n = 20;
if (n == 10){
  console.log("n = 10");
} else if (n == 90) {
  console.log("n = 90");
} else if (n == -3) {
  console.log("n = -3");
} else {
  console.log("none of the above");
}
```

```
let n = 20;
switch (n) {
  case 10:
    console.log("n = 10");
  case 90:
    console.log("n = 90");
  case -3:
    console.log("n = -3");
  default:
    console.log("none of the above");
}
```



Typeof operatorius

typeof operatoriaus grąžina argumento tipą.

Tai naudinga, kai norime skirtingai apdoroti skirtingų tipų vertes arba tiesiog norime atlikti greitą vertės tipo patikrinimą.

Palaikomos dvi sintaksės:
Kaip operatorius: *typeof x*
Kaip funkcija: *typeof(x)*

```
console.log(typeof 42);
// expected output: "number"

console.log(typeof 'blubber');
// expected output: "string"

console.log(typeof true);
// expected output: "boolean"

console.log(typeof undeclaredVariable);
// expected output: "undefined"
```