JS

1. Kintamieji

- Lokalūs kintamieji - tai kintamieji sukurti funkcijoje. Jie galioja sukurtoje funkcijoje.
- Globalūs kintamieji - tai kintamieji sukurti ne funkcijoje. Jie galioja visoje programoje.

Keyword	Scope	Redeclaration	Reassignment
var	function scope	yes	yes
let	block scope	no	yes
const	block scope	no	no

Kintamųjų tipai

- Primityvūs kintamieji:
  - Skaičiai / number (`0, 1, 1.3, NaN, Infinity`)
  - Raidžių eilutės/ strings (`'raidės', "eilutės"`)
  - Loginiai/ boolean (`true/false`)
  - `null`
  - `undefined`

```
let foo = 42; // foo is now a number
foo = "bar"; // foo is now a string
foo = true; // foo is now a boolean
```

3. Data Type

There are eight basic data types in JavaScript

List of Data Types

@geekymindsblog

Data Types	Description
Number	integers or floating-points
BigInt	integers of arbitrary length
String	series of one or more characters
Boolean	true or false
Null	unknown values
Undefined	unassigned values
Symbol	unique identifiers
Object	key:value pairs

Note: null and undefined are not the same thing. Null is empty or non-existent value that needs to be assigned to a variable. While, undefined is typically means a variable has been declared but not defined.

Number

NaN - (not a number) - yra speciali skaičiaus reikšmė, su kuria paprastai susiduriama, kai aritmetinės operacijos rezultato negalima išreikšti skaičiumi. Tai taip pat yra vienintelė „JavaScript“ vertė, kuri nėra lygi sau pačiam.

Infinity - begalybė yra globalaus objekto savybė. Kitaip tariant, tai yra globalus kintamasis.

```
console.log(Infinity);/* Infinity */
console.log(Infinity + 1); /* Infinity */
console.log(Math.pow(10, 1000)); /* Infinity */
console.log(Math.log(0)); /* -Infinity */
console.log(1 / Infinity); /* 0 */
console.log(1 / 0); /* Infinity */
```

Null

- „JavaScript“ programoje null yra speciali reikšmė, nurodanti tuščią arb a nežinomą reikšmę.
- let number = null;

Undefined

- Jei kintamasis deklaruojamas, bet reikšmė nepriskirta, to kintamojo reikšmė bus neapibrėžta.

```
let name;
console.log(name); // undefined

// Čia name yra niekam neprilygintas, todėl neveiks.
// Dažnai meta console, kai padarome klaidą priskyrime.
```

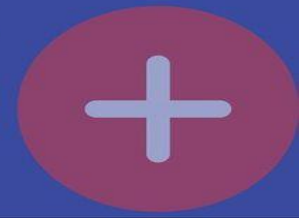
Masyvas

- Tipinė duomenų struktūra – masyvas, kuriame skaičiai (indeksai) susiejami su reikšmėmis. Galimybė susieti skaičių su reikšme yra ir kituose tipuose, bet masyvai turi ir specializuotą funkcionalumą (suliejimas, pridėjimas į galą ir pan.), taip pat masyvai turi masyvo dydį nusakantį atributą (`length`).

```
let mountains = ['Everest', 'Fuji', 'Nanga Parbat'];
console.log(mountains[0]); // 'Everest'
console.log(mountains[1]); // 'Fuji'
console.log(mountains[2]); // 'Nanga Parbat'
```

JAVASCRIPT ARRAY ADD ELEMENTS

REAL QUICK EXAMPLES



01 PUSH
ADD TO END OF ARRAY

```
var arr = ["One", "Two"];
arr.push("Three");
console.log(arr); ONE, TWO, THREE
```

02 UNSHIFT
ADD TO START OF ARRAY

```
var arr = ["One", "Two"];
arr.unshift("Three");
console.log(arr); THREE, ONE, TWO
```

03 ASSIGN
ACTS LIKE PUSH

```
var arr = ["One", "Two"];
arr[arr.length] = "Three";
console.log(arr); ONE, TWO, THREE
```

04 CONCAT
JOIN TWO ARRAYS

```
var arrA = ["One", "Two"];
var arrB = ["Three", "Four"];
var arrC = arrA.concat(arrB);
console.log(arrC); ONE, TWO, THREE, FOUR
```

05 SPLICE
FLEXIBLE USE

INSERT IN-BETWEEN

```
var arr = ["One", "Two"];
arr.splice(1, 0, "Three");
console.log(arr); ONE, TWO, THREE
```

INSERT MULTIPLE

```
var arr = ["One", "Two"];
arr.splice(1, 0, "Three", "Four");
console.log(arr); ONE, THREE, FOUR, TWO
```

VISIT CODE BOXX FOR MORE!

HTTPS://CODE-BOXX.COM/

tutorials | open source projects | code download

Objektas

- Objektai naudojami „daiktui“ jūsų kode pavaizduoti. Tai gali būti žmogus, automobilis, pastatas, knyga, žaidimo personažas – iš esmės viskas, kas yra sudaryta arba gali būti apibrėžta savybių rinkiniu. Objektuose šios charakteristikos vadinamos savybėmis, kurias sudaro raktas ir reikšmė.

```
// Basic object syntax
var object = {
key: 'value'
};
```

```
// Example 'person' object
var person = {
name: 'Zac',
age: 33,
likesCoding: true
};
```

2. Operatoriai

OPERACIJA	PAVYZDYS	KĄ REIŠKIA
sudėtis	kint1 + kint2	Sudeda du kintamuosius
atimtis	kint1 - kint2	Iš kint1 atima kint2
daugyba	kint1 * kint2	Sudaugina du kintamuosius
dalyba	kint1 / kint2	Padalina kint1 iš kint2
liekana	kint1 % kint2	Padalina kint1 iš kint2 ir gražina liekaną
kėlimas laipsniu	Math.pow(kint1, laips)	Pakelia $kint1^{laipsniu}$
apvalinimas	Math.round(kint1)	Suapvalina kint1 iki sveiko skaičiaus

Category	Operator	Name/Description	Example	Result
Arithmetic	+	Addition	3+2	5
	-	Subtraction	3-2	1
	*	Multiplication	3*2	6
	/	Division	10/5	2
	%	Modulus	10%5	0
	++	Increment and then return value	X=3; ++X	4
		Return value and then increment	X=3; X++	3
	--	Decrement and then return value	X=3; --X	2
		Return value and then decrement	X=3; X--	3
Logical	&&	Logical “and” evaluates to true when both operands are true	3>2 && 5>3	False
	||	Logical “or” evaluates to true when either operand is true	3>1 || 2>5	True
	!	Logical “not” evaluates to true if the operand is false	3!=2	True
Comparison	==	Equal	5==9	False
	!=	Not equal	6!=4	True
	<	Less than	3<2	False
	<=	Less than or equal	5<=2	False
	>	Greater than	4>3	True
	>=	Greater than or equal	4>=4	True
String	+	Concatenation(join two strings together)	“A”+”BC”	ABC

3. Sąlygos

If

Else/if

Else

Switch

Ternary

AND

IF

- Teiginys `if(...)` įvertina sąlygas esančias tarp skliaustelių ir jeigu rezultatas yra `true` tada įvykdo kodų rinkinį.

```
if (year == 2015) {
alert( "Jūs teisi(-us)!" ); alert( "Jūs
protinga(-as)!" );
}
```

```
let amzius = 15;
if(amzius >= 18) {
    console.log("Esipilnametis");
}
```

Else if

- Jeigu pirmoji sąlyga (IF) yra tiesa, tada vykdoma else if tolimesniam patikrinimui,

```
let value = prompt('Įveskite skaičių', 0);

if (value > 0) {
alert( 1 );
 } else if (value < 0)
{ alert( -1 );
} else {
alert( 0 ); }
```

```
let year = prompt('Kuriais metais buvo išleista ECMAScript-
2015 specifikacija?', '');
if (year < 2015) {
alert( 'Per anksti...' );
else if (year > 2015)
{ alert( 'Per vėlai' );
} else {
alert( 'Būtent!' ); }
```

Else

- Į `if` teiginį gali būti įterptas neprivalomas “else” blokas. Jis vykdomas, kai sąlyga yra “falsy”.

```
let year = prompt('Kuriais metais buvo išleista ECMAScript-2015
specifikacija?', '');
if (year == 2015) {
alert( 'Jūs atspėjote!' );
} else {
alert( 'Kaip galėjote taip suklysti?' ); // bet kuriai kitai vertei
išskyrus 2015 }
```

Switch

- Teiginys `switch` gali pakeisti daugybinius `if` patikrinimus.
- Jis suteikia lengviau apibūdinamą kelią palyginti vertes su įvairiais variantais.
- Teiginys `switch` turi vieną ir daugiau `case` (bylos) blokų ir numatytąjį pasirinkimą.

```
let megstamiausiaSpalva = 'geltona';
switch(megstamiausiaSpalva){
    case 'geltona':
    case 'raudona':
        console.log('megsti ryskia');
        break;
    case 'melyna':
    case 'ruda':
        console.log('megsti tamsias spalvas');
        break;
```

```
let a = 2 + 2; switch (a) {
 case 3:
alert( 'Per mažas' );
break;
case 4:
alert( 'Kaip tik!' );
break;
case 5:
alert( 'Per didelis' );
break;
default: alert( "Tokios vertės nežinau" ); }
```

Ternary

AND

4. Ciklai

For

While

Do while

For

- Ciklas `for` yra kiek sudėtingesnis, bet jis tuo pačiu yra dažniausiai naudojamas ciklas.
- Jis atrodo taip:

```
for (begin; condition; step)
```

PRADŽIA	LET I = 0	ĮVYKDOMAS VIENĄ KARTĄ PRADEDANT CIKLĄ.
salyga	i < 3	Patikrinama prieš kiekvieną ciklo iteraciją. Jeigu netiesa, ciklas sustoja.
korpusas	alert(i)	Įvykdomas vėl ir vėl kol sąlyga yra truthy.
žingsnis	i++	Įvykdomas po korpuso per kiekvieną iteraciją.

Pvz.

- Išmokime šių dalių reikšmę su pavyzdžiais. Ciklas žemiau paleidžia `alert(i)` kol `i` yra nuo `0` iki (bet neįskaitant) `3`:

```
for (let i = 0; i < 3; i++) { // parodo 0, tada 1, tada 2
alert(i); }
```

```
// for (let i = 0; i < 3; i++) alert(i)
// pradedamas vykdymas
let i = 0
// jeigu sąlyga → paleisti korpusą ir paleisti žingsnį
if (i < 3) { alert(i); i++ }
// jeigu sąlyga → paleisti korpusą ir paleisti žingsnį
if (i < 3) { alert(i); i++ }
// jeigu sąlyga → paleisti korpusą ir paleisti žingsnį
if (i < 3) { alert(i); i++ }
// ...pabaiga, nes dabar i == 3
```

While

- `while` -- Sąlyga patikrinima prieš kiekvieną iteraciją.
- Kol `sąlyga` yra truthy, `kodas` iš ciklo rinkinio yra įvykdomas.

```
while (sąlyga) {
// kodas
// taip vadinamas “ciklo korpusas” (ang. “loop body”)
}
```

Pavyzdžiui, ciklas žemiau atiduoda i kol i < 3:

```
let i = 0;
while (i < 3) { // parodo 0, tada 1, tada 2
alert( i );
i++; }
```

Do while

- Sąlygos patikrinimas gali būtų perkeltas *žemiau* ciklo korpuso naudojant sintaksę `do..while`:

```
do {
 // ciklo korpusas
} while (sąlyga);
```

Ciklas visų pirma įvykdys korpusą, tada patikrins sąlygą ir kol ji yra truthy, įvykdys vėl ir vėl.

```
let i = 0;
do {
  alert( i );
  i++;
} while (i < 3);
```

Tokia sintaksė turėtų būti naudojama kai norite, kad ciklo korpusas būtų įvykdytas bent vieną kartą nepaisant to ar jo sąlyga yra truthy. Dažniausiai vis dėlto naudojama kita forma: while(…) {…}.

5. Funkcijos

- Funkcijos deklaravimas:

```
function showMessage() {
alert( 'Labas visiems!' );
}
```

Mūsų naująją funkciją galima iškviesti jos pavadinimu: showMessage().

Lokaliniai kintamieji

Veikia tik funkcijos viduje esantys LET

```
function showMessage() {

 let message = "Sveiki, aš esu JavaScript!"; // lokalinis kintamasis


  alert( message );
}

showMessage(); // Sveiki, aš esu JavaScript!

alert( message ); // <-- bus sukelta klaida, nes kintamasis matomas tik
funkcijos viduje.
```

Išoriniai kintamieji

Funkcija turi prieigą prie išorinių kintamųjų, pavyzdžiui:

```
let userName = 'Jonas';

function showMessage() {
  let message = 'Sveiki, ' + userName;
  alert(message);
}

showMessage(); // Sveiki, Jonas
```

Return

- Funkcija gali grąžinti rezultatą, kuris bus perduotas ją iškvietusiam kodui.

```
function sudetis(nr1, nr2){
    return nr1 + nr2; //arba let atsakymas = nr1
+ nr2 // return atsakymas;
    // uz return console.log neivyks
}
```

```
function min(a, b) {
if (a < b) {
return a;
} else {
return b; } }
```