**მონაცემთა ტიპები და ოპერატორები**

1. **მონაცემთა ტიპები, typeof**

**Number -** რიცხვი - გამოიყენება როგორც მთელი ასევე არამთელი მნიშვნელობებისთვის

var n = 123;

n = 12.345;

*სპეციალური მნიშვნელობები:* **Infinity, NaN**

alert( 1 / 0 ); // Infinity

alert( "რიხცვი" \* 2 ); // NaN, შეცდომა

**String -** სტრიქონი

var str = "მარტივი წინადადება";

str = 'ერთმაგი ბრჭყალების გამოყენებაც დასაშვებია';

**Boolean -** ლოგიკური ტიპი

var checked = true; // მაგალითად ფორმის ველი (ჩეკბოქსი) მონიშნულია

checked = false; // ფორმის ველი არ არის მონიშნული

**Null -** სპეციალური მნიშვნელობა - აღნიშნავს რომ მნიშევნელობა არ არის ცნობილი

var age = null;

**Undefined** - სპეციალური მნიშვენლობა, აღნიშნავს რომ მნიშვნელობა არ არის მინიჭებული

var x;

alert( x ); // გამოდის "undefined"

შეიძლება undefined პირდაპირი მინიჭებაც, თუმცა აზრს მოკლებულია და იშვიათად გამოიყენება რადგან უცნობი მნიშვნელობის აღსანიშნად არის null

var x = 123;

x = undefined;

alert( x ); // "undefined"

**Object** - ობიექტები, განსხვავდება წინა ჩამოთვლილი ტიპებისგან იმით რომ არ არის პრიმიტიული ტიპი. გამოიყენება კოლექციების და კომპლექსური მონაცემების ერთეულების აღსაწერად.

var user = { name: "დავითი" };

**ოპერატორი typeof**

სინტაქსურად აქვს 2 ფორმა

1. ოპერატორის ფორმა ფრჩხილების გარეშე: typeof x
2. ფუნქციის ფორმა ფრჩხილებით: typeof(x)

შედეგად მიიღება სტრიქონი რომელშიც მოცემულია მონაცემთა ტიპი

typeof undefined // "undefined"

typeof 0 // "number"

typeof true // "boolean"

typeof "foo" // "string"

typeof {} // "object"

typeof null // "object" (1)

typeof function(){} // "function" (2)

**2. ძირითადი ოპერატორები**

**სტრიქონების დამატება, ბინარული +**

+ გამოიყენება უფრო ხშირად როცხვებზე თუმცა სტრიქონებზე გამოყენებისას ხდება მათი შეწებება (კონკატენაცია)

var a = "js" + "მარტივია";

alert( a ); // jsმარტივია

**სხვა არითმეტიკული ოპერატორები მუშაობენ მხოლოდ რიცხვებზე**

alert( 2 - '1' ); // 1

alert( 6 / '2' ); // 3

**უნარული პლიუსი**

რიცხვებზე გამოყენებისას არაფერს არ ცვლის

alert( +1 ); // 1

alert( +(1 - 2) ); // -1

თუმცა გამოიყენება სტრიქონების რიცხვებად გარდაქმნისთვის

ბინარული უბრალოდ აწებებს:

var apples = "2";

var oranges = "3";

alert( apples + oranges ); // "23", ბინარული პლიუსი აწებებს სტრიქონებს

უნარული:

var apples = "2";

var oranges = "3";

alert( +apples + +oranges ); // 5, რიცხვი, ორივე ოპერანდი ჯერ გარდაიქმნა რიცხვებად

**მინიჭების ოპერატორი**

მარტივი მინიჭება

var x = 2 \* 2 + 1;

alert( x ); // 5

შესაძლებელია ჯაჭვური მინიჭებაც

var a, b, c;

a = b = c = 2 + 2;

alert( a ); // 4

alert( b ); // 4

alert( c ); // 4

ოპერატორი = როგორც ყველა სხვა ოპერატორი აბრუნებს მნიშვნელობას, ჯერ მიანიჭებს და მერე აბრუნებს მინიჭებულს, ამიტომ შესაძლებელია მისი გამოყენება უფრო რთული გამოსახულების ნაწილად

var a = 1;

var b = 2;

var c = 3 - (a = b + 1);

alert( a ); // 3

alert( c ); // 0

**ნაშთის გამოთვლის ოპერატორი %**

alert( 5 % 2 ); // 1, 5-ის 2-ზე გაყოფის ნაშთი

alert( 8 % 3 ); // 2, 8-ის 3-ზე გაყოფის ნაშთი

alert( 6 % 3 ); // 0, 6-ის 3-ზე გაყოფის ნაშთი

**ინკრემენტი/დეკრემენტი: ++, --**

ინკრემენტი

var i = 2;

i++; // i = i + 1 შემოკლებული ვერსია

alert(i); // 3

დეკრემენტი

var i = 2;

i--; // შემოკლებული ვერსია, იგივე რაც i = i - 1.

alert(i); // 1

არსებობს ამ ოპერატორების პრეფიქსული ფორმაც

var i = 1;

var a = ++i; // (\*)

alert(a); // 2

*განსხვავება ამ ორ ფორმას შორის ჩანს თუ კი ინკრემენტის შედეგის გამოყენება გვინდა იქვე გამოსახულებაში. პოსტფიქსული აბრუნებს ძველ მნიშვნელობას სანამ გაიზრდებობდა, პრეფიქსული უკვე გაზრდილს*

var i = 1;

var a = i++; // (\*)

alert(a); // 1

**შემოკლებული არითმეტიკული ოპერატორები**

გრძელი ვარიანტი

var n = 2;

n = n + 5;

n = n \* 2;

შემოკლებული ვარიანტი

var n = 2;

n += 5; // n=7 (იგივეა რაც n = n + 5)

n \*= 2; // n=14 (იგივეა რაც n = n \* 2)

alert( n ); // 14

**ოპერატორი ,**

გამოითვლება გამოსახულებები და იგნორირდება ყველა ბოლოს გარდა

var a = (5, 6);

alert( a );

გამოიყენება კოდის შესამცირებლად

// სამი ოპერაცია ერთ სტრიქონში

for (a = 1, b = 3, c = a\*b; a < 10; a++) {

...

}

**დავალებები:**

შეასრულეთ შემდეგი კოდი და ყოველ კონკრეტულ შემთხვევაში ახსენით როგორ მიიღება შედეგი რომელიც პროგრამას გამოაქვს

var a = 1, b = 1, c, d;

c = ++a; alert(c); // 2

d = b++; alert(d); // 1

c = (2+ ++a); alert(c); // 5

d = (2+ b++); alert(d); // 4

alert(a); // 3

alert(b); // 3

რისი ტოლი იქნება x ? შეასრულეთ პროგრამა და შეამოწმეთ თქვენი პასუხი

var a = 2;

var x = 1 + (a \*= 2);

**3. შედარების ოპერატორები, ლოგიკური მნიშვნელობები**

**რიცხვების შედარება**

alert( 2 > 1 ); // true

alert( 2 == 1 ); // false

alert( 2 != 1 ); // true

**ლოგიკურ მნიშვნელობებთან მუშაობა**

var a = true; // პირდაპირი მინიჭება

var b = 3 > 4; // შედარების შედეგის მინიჭება

alert( b ); // false

alert( a == b ); // (true == false) გამოვა false

**განსხვავებული ტიპების შედარება**

alert( '2' > 1 ); // true, დარდება როგორც 2 > 1

alert( '01' == 1 ); // true, დარდება როგორც 1 == 1

alert( false == 0 ); // true, false ხდება 0

alert( true == 1 ); // true, რადგან true ხდება 1.

მკაცრი ტოლობა

alert( 0 === false ); // false, რადგან ტიპები განსხვავებულია