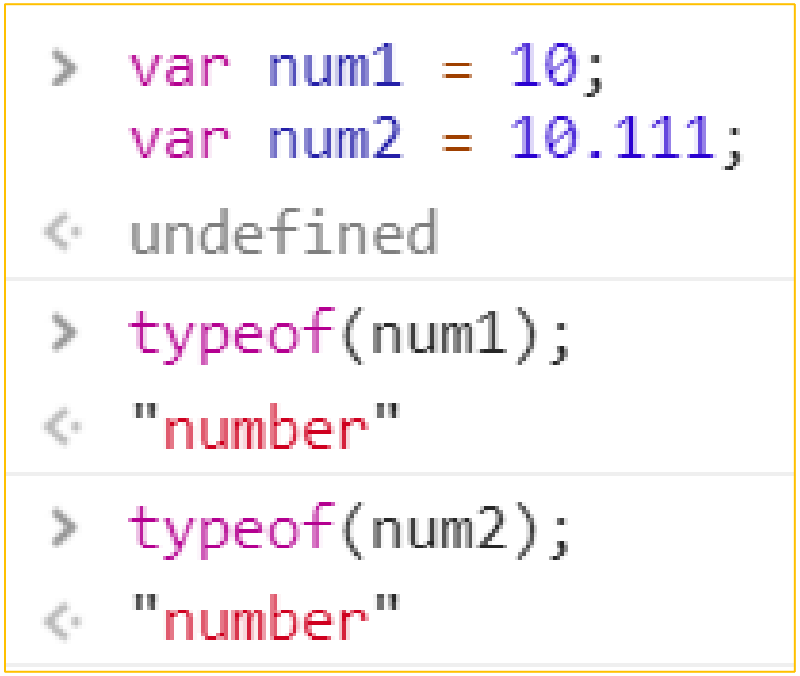
JS - DUOMENŲ TIPAI

Data Types

- Number
- Boolean
- Null
- Undefined
- String

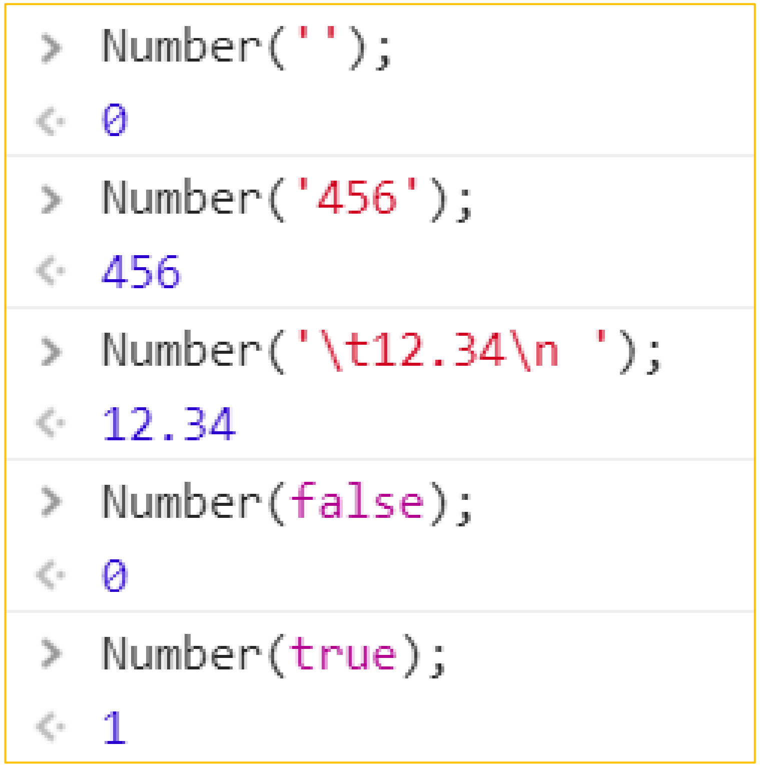
- Object

Numbers

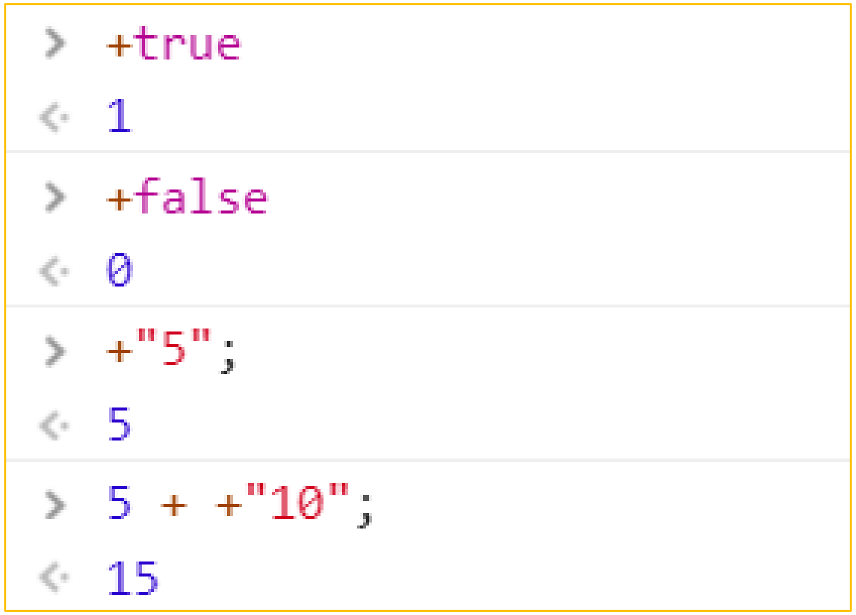


```
> var num1 = 10;
  var num2 = 10.111;
<· undefined
> typeof(num1);
<· "number"
> typeof(num2);
<· "number"
```

Converting to Number



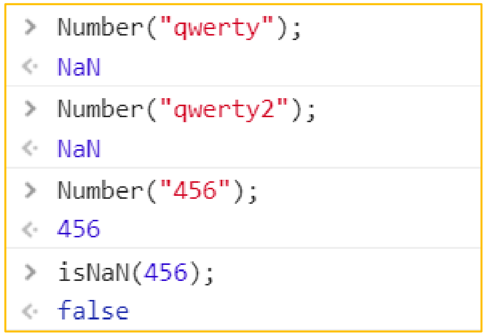
```
> Number('');
<· 0
> Number('456');
<· 456
> Number('\t12.34\n ');
<· 12.34
> Number(false);
<· 0
> Number(true);
<· 1
```



```
> +true
<· 1
> +false
<· 0
> +"5";
<· 5
> 5 + +"10";
<· 15
```

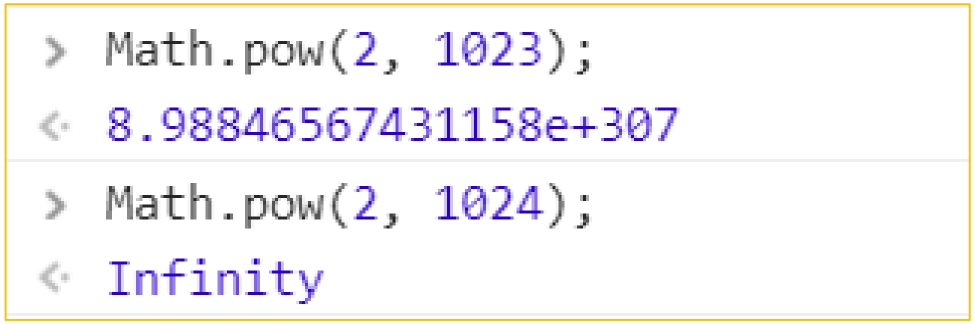
Special Number Values (error)

- **NaN**:
  - not a number;



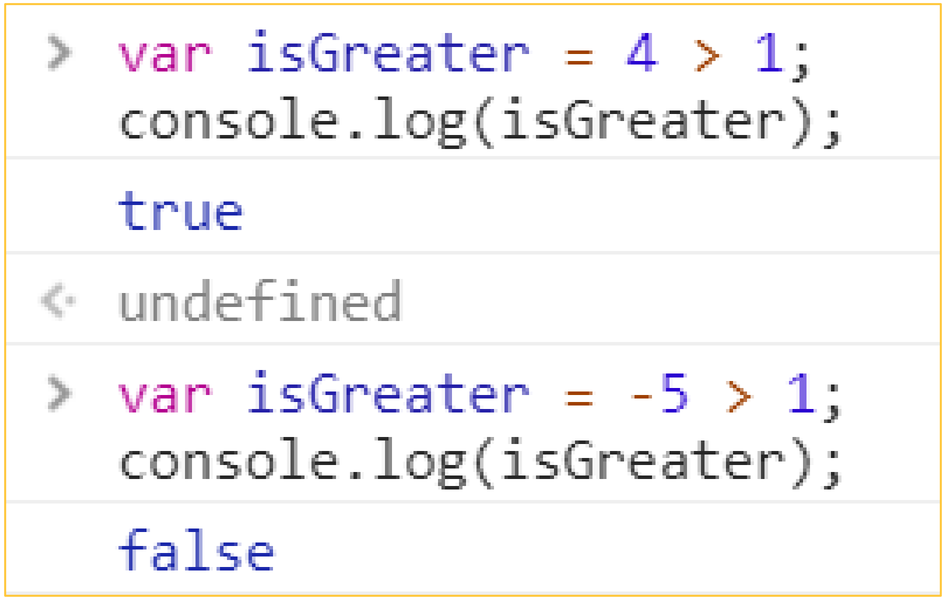
```
> Number("qwerty");
<· NaN
> Number("qwerty2");
<· NaN
> Number("456");
<· 456
> isNaN(456);
<· false
```

- **Infinity**:
  - a number can’t be represented because its magnitude is too large;
  - a division by zero has happened.



```
> Math.pow(2, 1023);
<· 8.98846567431158e+307
> Math.pow(2, 1024);
<· Infinity
```

Booleans



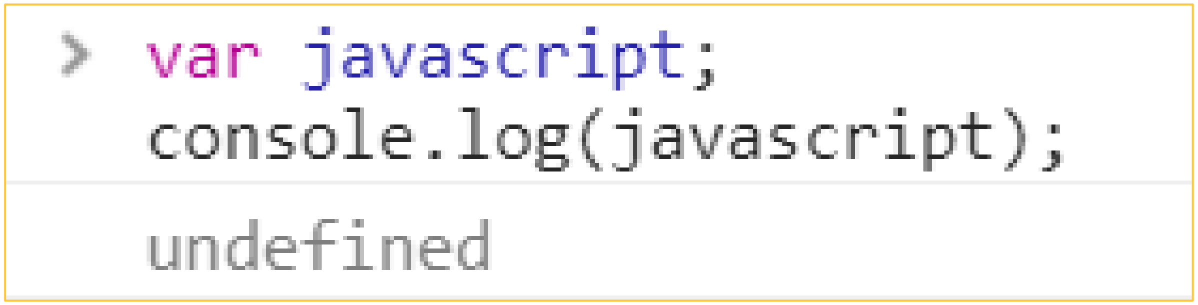
```
> var isGreater = 4 > 1;
  console.log(isGreater);
  true
< undefined
> var isGreater = -5 > 1;
  console.log(isGreater);
  false
```

The “null”

- The special null value does not belong to any type;
- In JavaScript null is not a “reference to a non-existing object” or a “null pointer” like in some other languages.
- It’s just a special value which has the sense of “nothing”, “empty” or “value unknown”.

The “undefined”

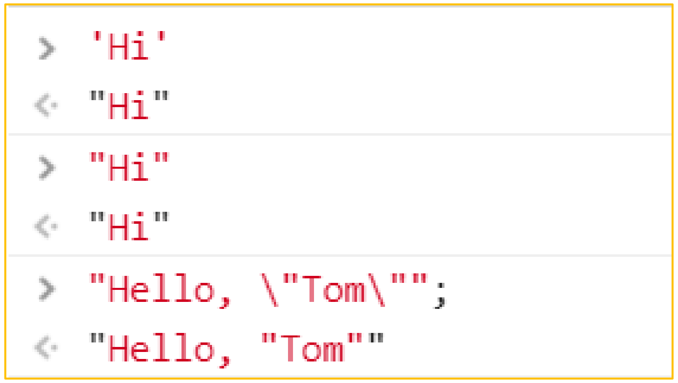
- The meaning of undefined is “value is not assigned”;
- If a variable is declared, but not assigned, then its value is exactly ***undefined***.



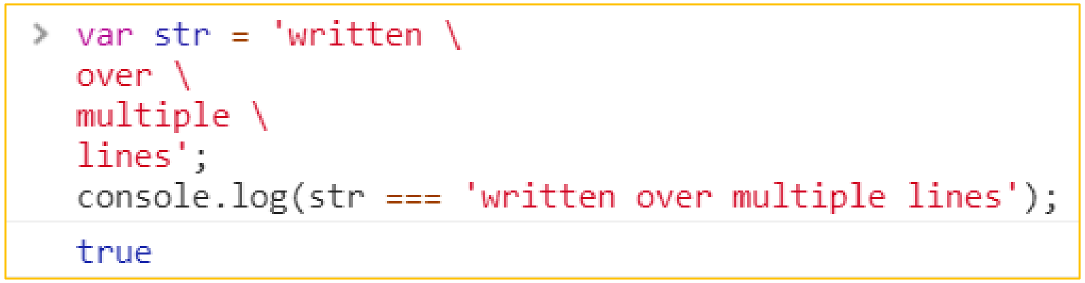
```
> var javascript;
  console.log(javascript);
  undefined
```

Strings (1)

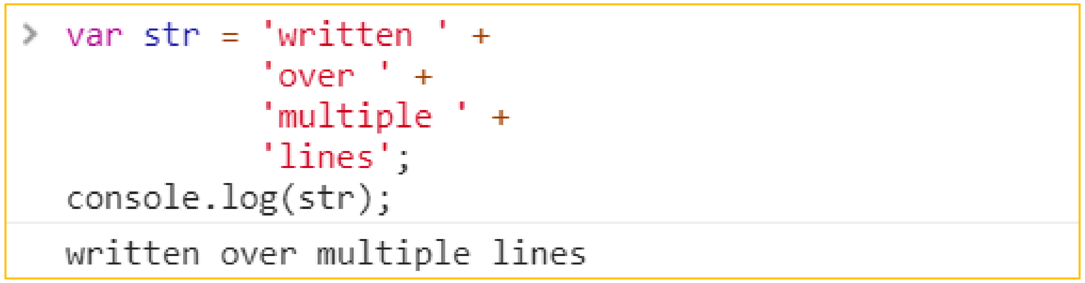
- Both single and double quotes can be used to delimit string literals.



```
> 'Hi'
< "Hi"
> "Hi"
< "Hi"
> "Hello, \"Tom\"";
< "Hello, "Tom""
```



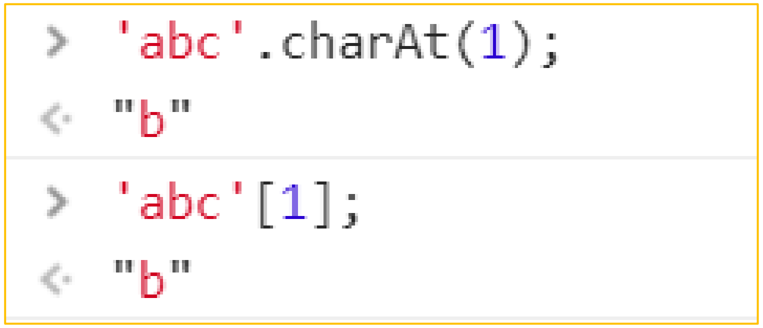
```
> var str = 'written \
  over \
  multiple \
  lines';
  console.log(str === 'written over multiple lines');
  true
```



```
> var str = 'written ' +
            'over ' +
            'multiple ' +
            'lines';
  console.log(str);
  written over multiple lines
```

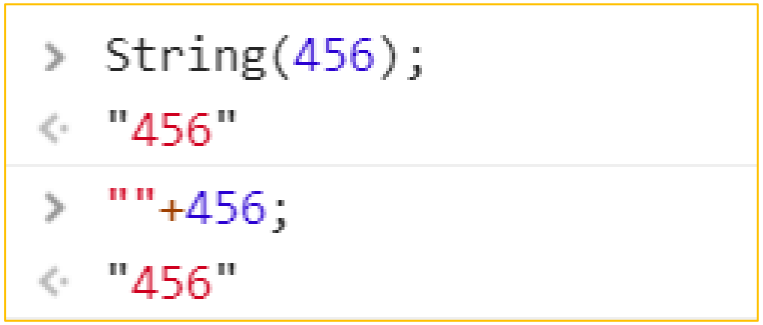
Strings (2)

- Character Access



```
> 'abc'.charAt(1);
<· "b"
> 'abc'[1];
<· "b"
```

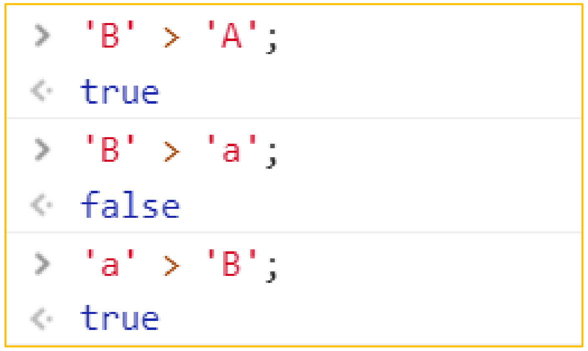
- Converting to String



```
> String(456);
<· "456"
> ""+456;
<· "456"
```

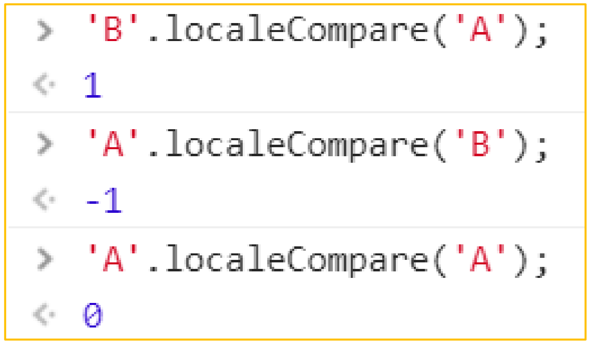
Strings (3)

- There are two ways of comparing strings:
  - comparison operators: <, >, ===, <=, >=.



```
> 'B' > 'A';
<· true
> 'B' > 'a';
<· false
> 'a' > 'B';
<· true
```

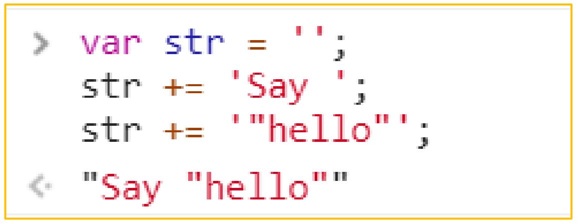
  - localeCompare(other)



```
> 'B'.localeCompare('A');
<· 1
> 'A'.localeCompare('B');
<· -1
> 'A'.localeCompare('A');
<· 0
```

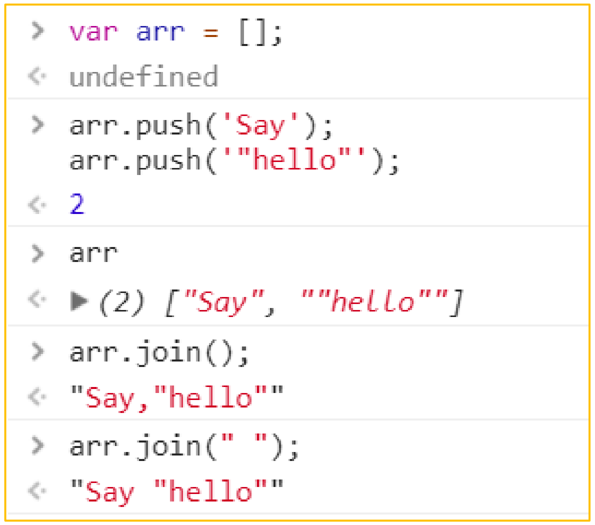
Strings (4) Concatenating Strings

- The Plus (+) Operator



```
> var str = '';
  str += 'Say ';
  str += '"hello"';
< "Say "hello""
```

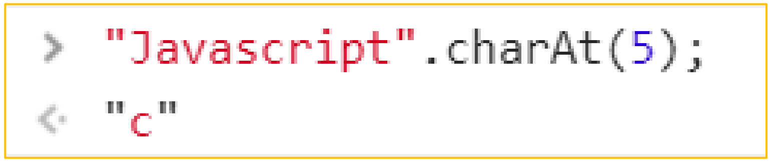
- Joining an Array of String Fragments



```
> var arr = [];
< undefined
> arr.push('Say');
  arr.push('"hello"');
< 2
> arr
< ▶ (2) ["Say", ""hello""]
> arr.join();
< "Say,"hello""
> arr.join(" ");
< "Say "hello""
```

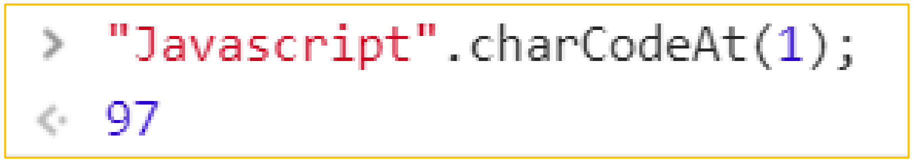
Strings (5)

- charAt(pos);



```
> "Javascript".charAt(5);
<· "c"
```

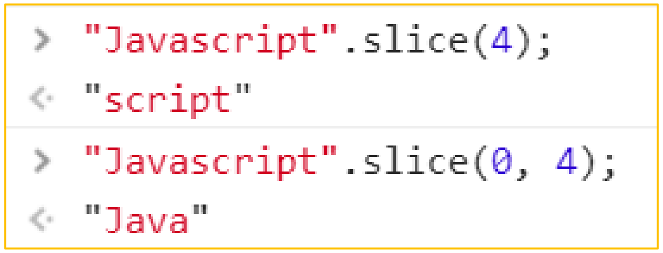
- charCodeAt(pos);



```
> "Javascript".charCodeAt(1);
<· 97
```

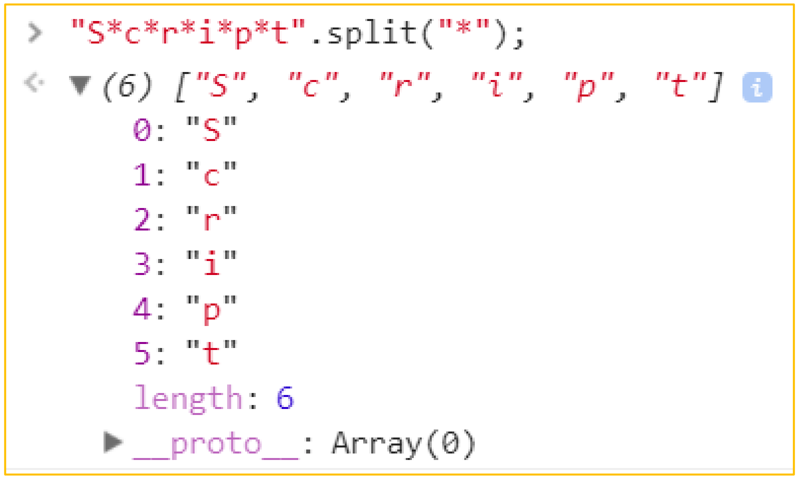
Strings (6)

- slice(start, end?), substring(start, end?);

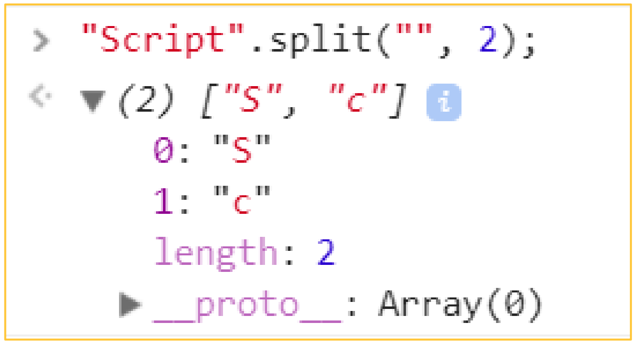


```
> "Javascript".slice(4);
<· "script"
> "Javascript".slice(0, 4);
<· "Java"
```

- split(separator?, limit?);



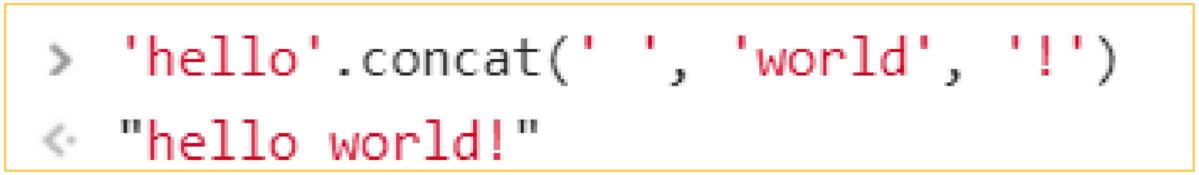
```
> "S*c*r*i*p*t".split("*");
<· ▼(6) ["S", "c", "r", "i", "p", "t"] i
      0: "S"
      1: "c"
      2: "r"
      3: "i"
      4: "p"
      5: "t"
      length: 6
    ▶ __proto__: Array(0)
```



```
> "Script".split("", 2);
<· ▼(2) ["S", "c"] i
      0: "S"
      1: "c"
      length: 2
    ▶ __proto__: Array(0)
```

Strings (7)

- concat(str1?, str2?, ...);

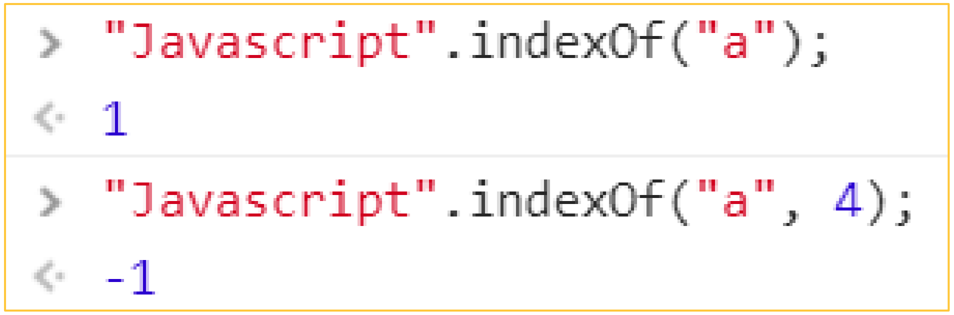


```
> 'hello'.concat(' ', 'world', '!')
<· "hello world!"
```

- toLowerCase();
- toUpperCase();

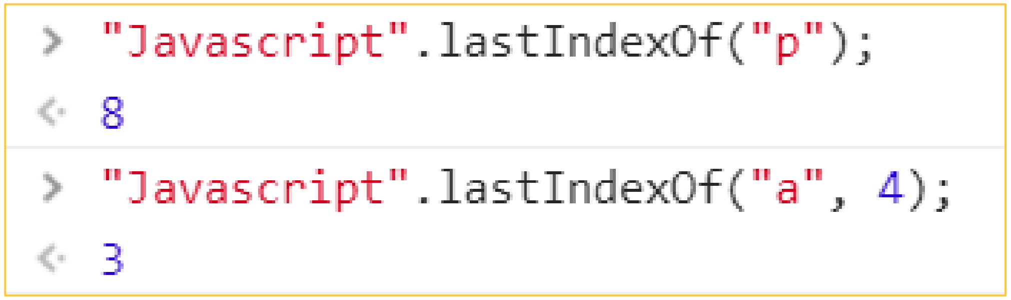
Strings (8)

- indexOf(searchString, position?);



```
> "Javascript".indexOf("a");
<· 1
> "Javascript".indexOf("a", 4);
<· -1
```

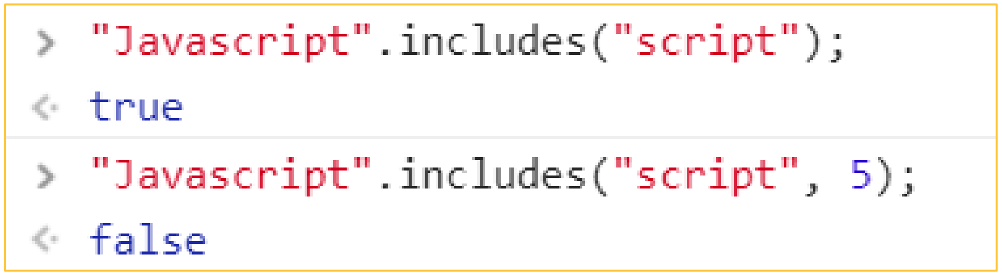
- lastIndexOf(searchString, position?)



```
> "Javascript".lastIndexOf("p");
<· 8
> "Javascript".lastIndexOf("a", 4);
<· 3
```

Strings (9)

- includes(*searchString*, position?);

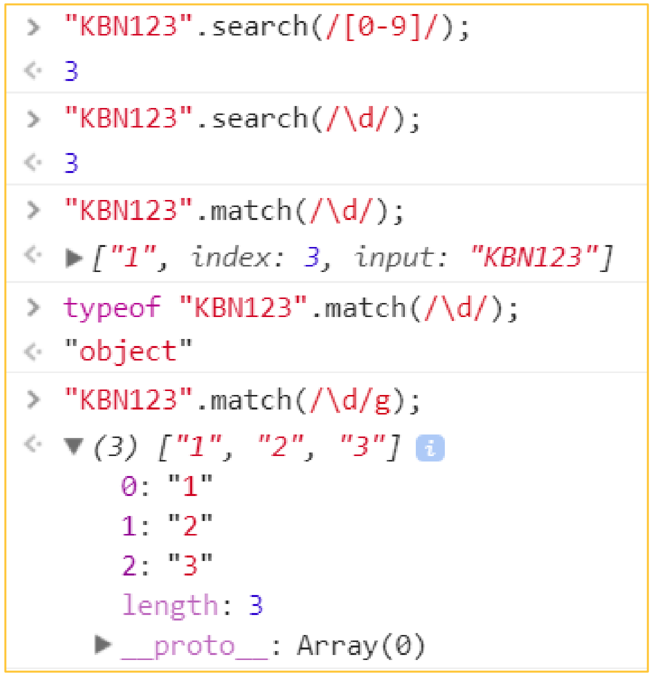


```
> "Javascript".includes("script");
< true
> "Javascript".includes("script", 5);
< false
```

- startsWith(searchString, position?);
- endsWith(*searchString*, length?);

Strings (10)

- search(regexp);
  - [regexp - regular expression]()
- match(regexp);

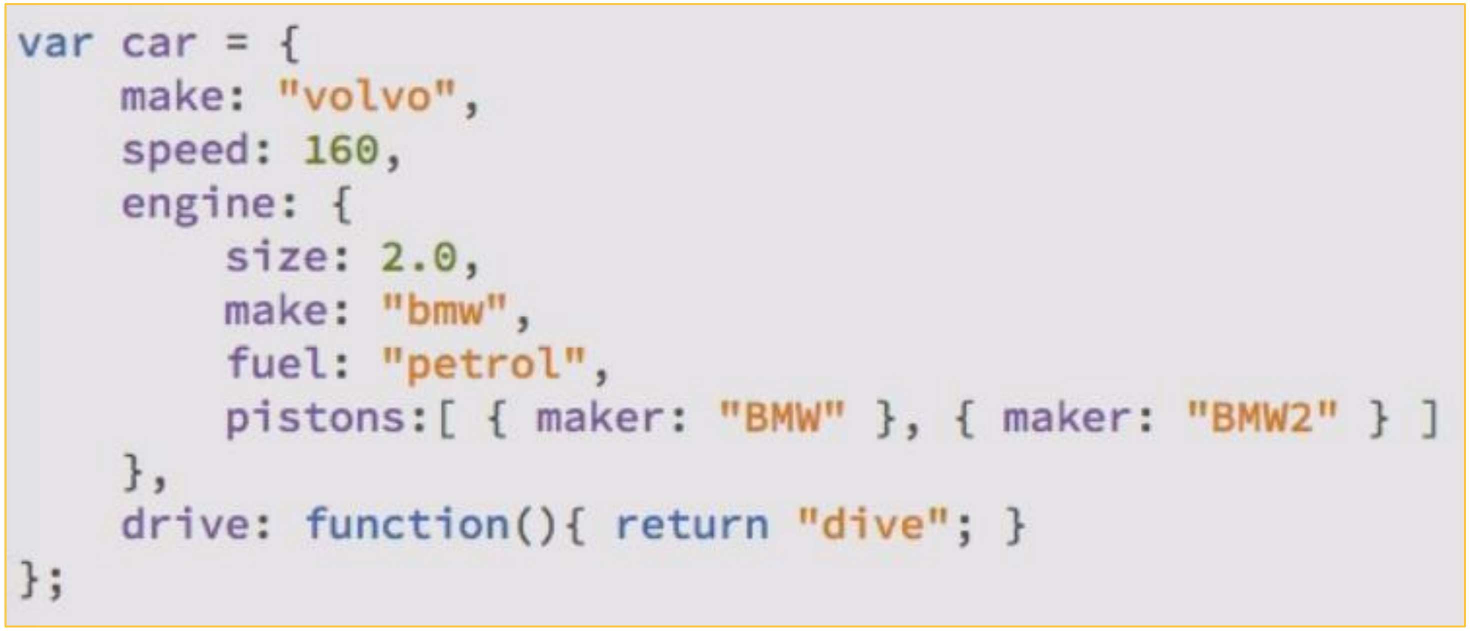


```
> "KBN123".search(/[0-9]/);
<· 3
> "KBN123".search(/\d/);
<· 3
> "KBN123".match(/\d/);
<· ▶["1", index: 3, input: "KBN123"]
> typeof "KBN123".match(/\d/);
<· "object"
> "KBN123".match(/\d/g);
<· ▼(3) ["1", "2", "3"] i
      0: "1"
      1: "2"
      2: "3"
      length: 3
    ▶ __proto__: Array(0)
```

No ***character*** type

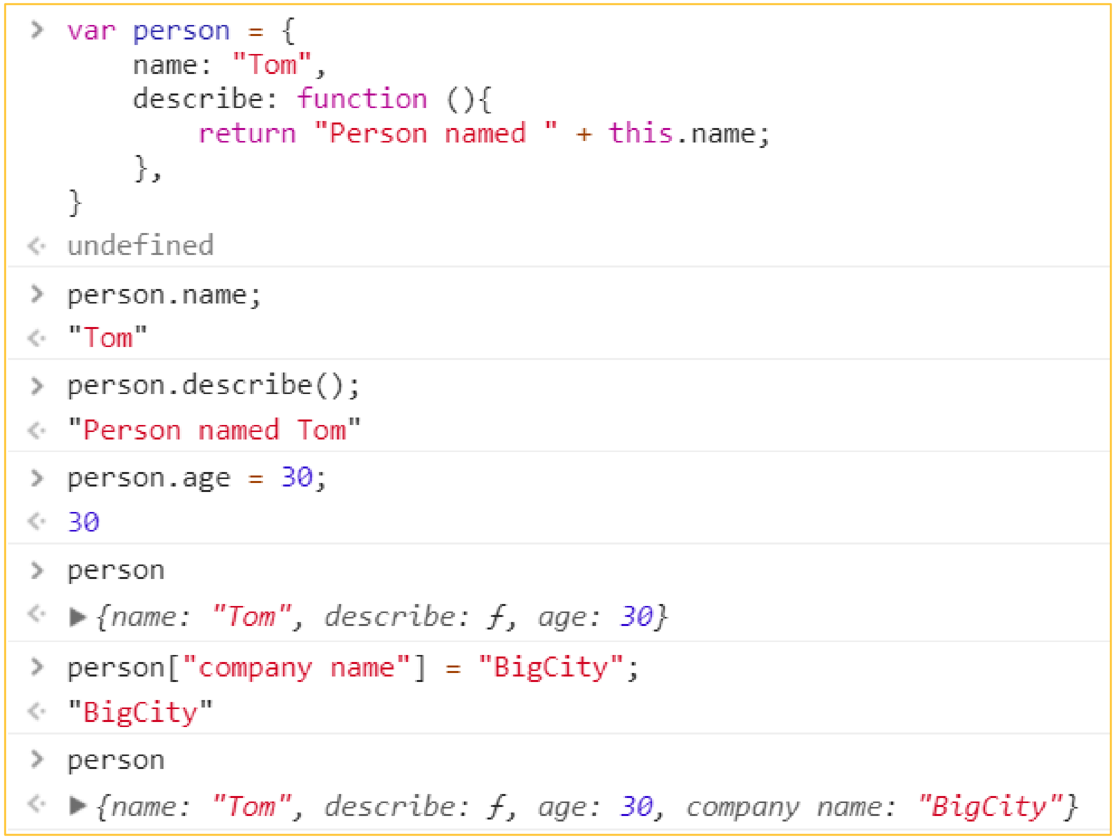
- There is no ***character*** type.
- There's only one type: string. A string may consist of only one character or many of them.

Objects (1)

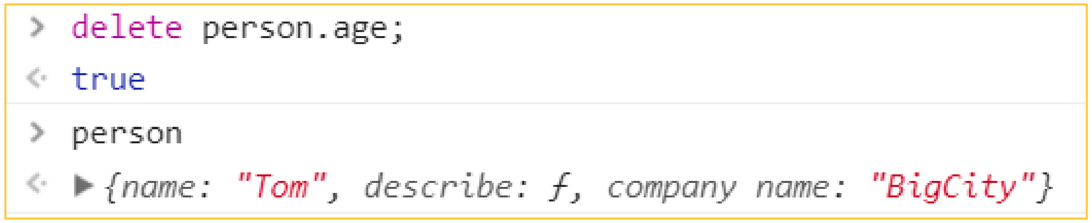


```
var car = {
    make: "volvo",
    speed: 160,
    engine: {
        size: 2.0,
        make: "bmw",
        fuel: "petrol",
        pistons:[ { maker: "BMW" }, { maker: "BMW2" } ]
    },
    drive: function(){ return "dive"; }
};
```

Objects (2)

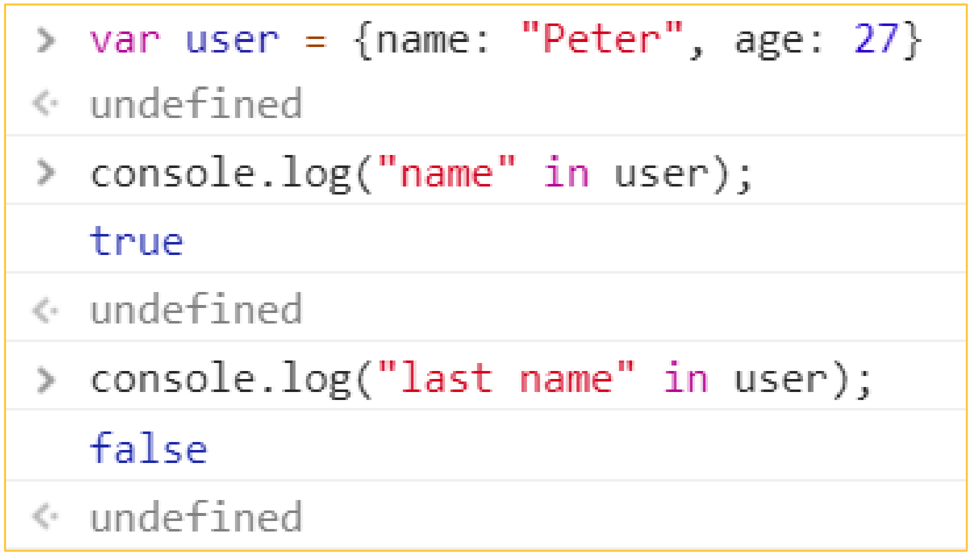


```
> var person = {
      name: "Tom",
      describe: function (){
          return "Person named " + this.name;
      },
  }
<· undefined
> person.name;
<· "Tom"
> person.describe();
<· "Person named Tom"
> person.age = 30;
<· 30
> person
<· ▶{name: "Tom", describe: ƒ, age: 30}
> person["company name"] = "BigCity";
<· "BigCity"
> person
<· ▶{name: "Tom", describe: ƒ, age: 30, company name: "BigCity"}
```



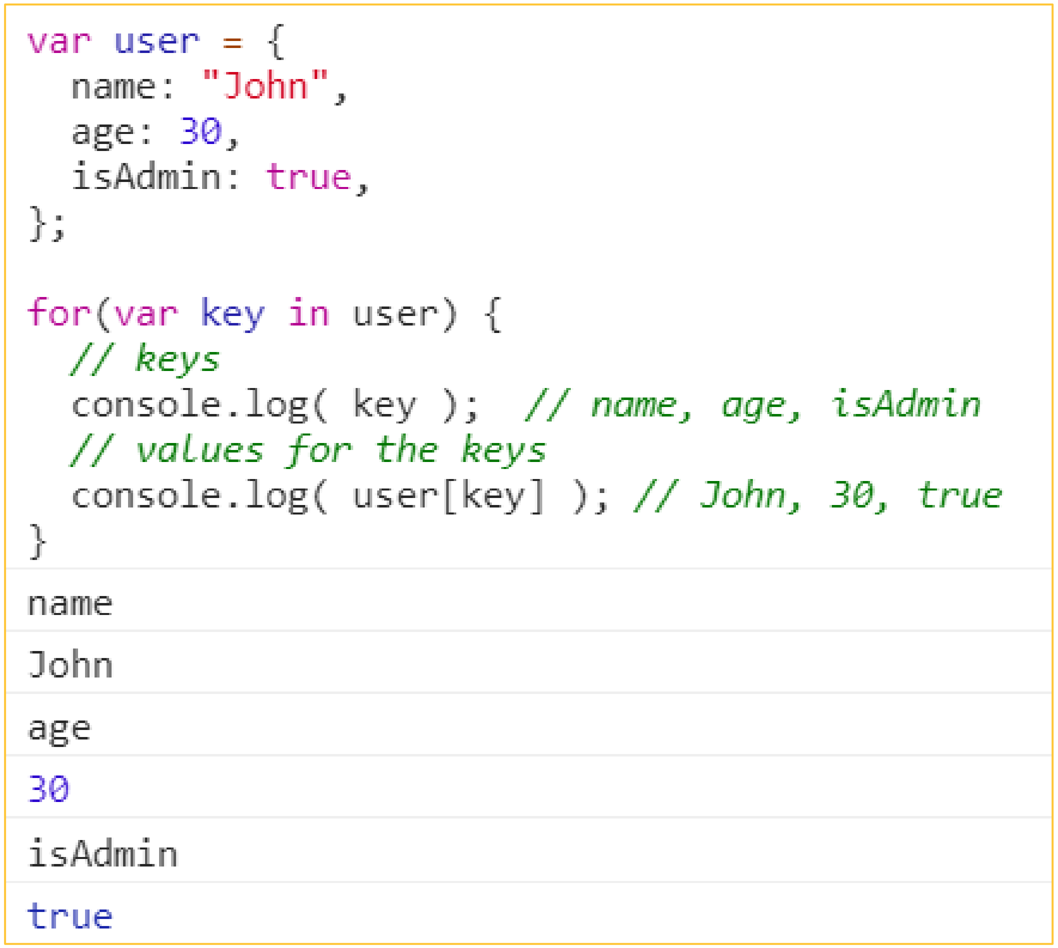
```
> delete person.age;
<· true
> person
<· ▶{name: "Tom", describe: ƒ, company name: "BigCity"}
```

Objects (3) **in**



```
> var user = {name: "Peter", age: 27}
< undefined
> console.log("name" in user);
  true
< undefined
> console.log("last name" in user);
  false
< undefined
```

Objects (3) **for...in**



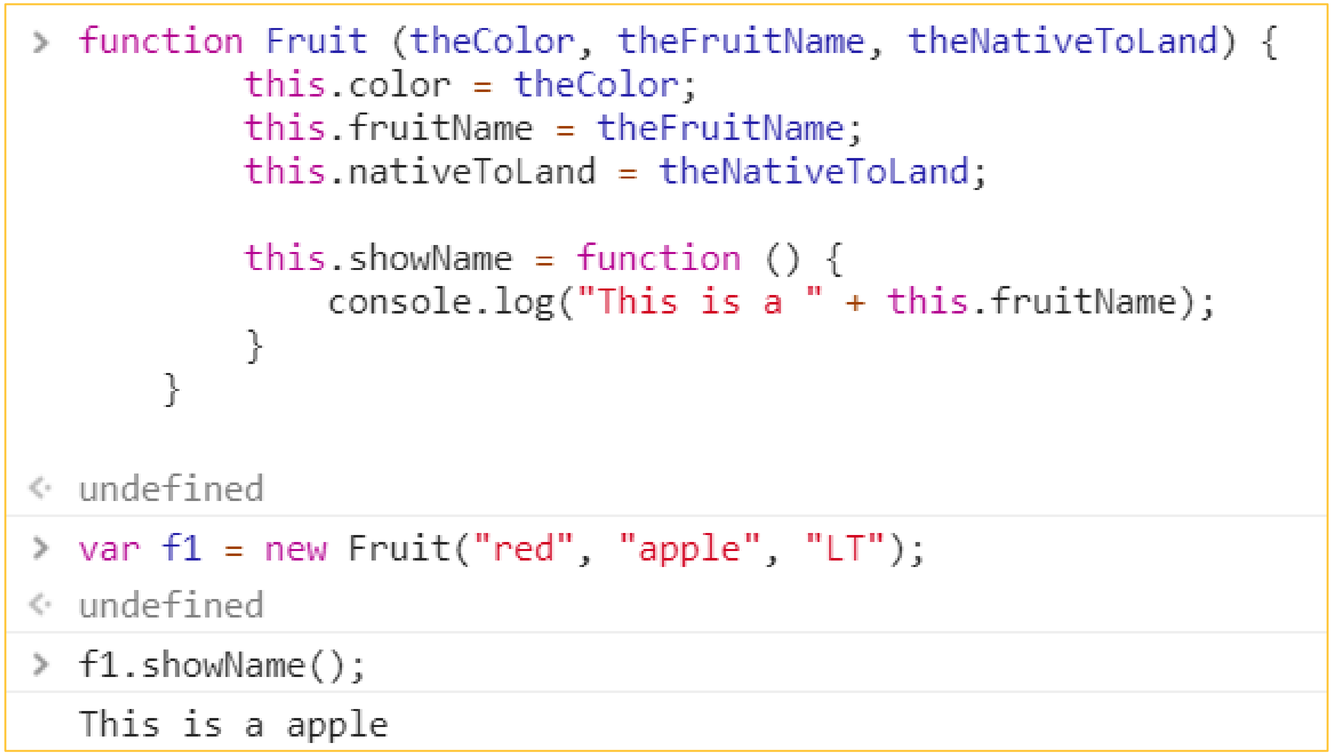
```
var user = {
  name: "John",
  age: 30,
  isAdmin: true,
};

for(var key in user) {
  // keys
  console.log( key );  // name, age, isAdmin
  // values for the keys
  console.log( user[key] ); // John, 30, true
}
```

```
name
John
age
30
isAdmin
true
```

Objects (4)

- Constructor Pattern for Creating Objects



```
> function Fruit (theColor, theFruitName, theNativeToLand) {
          this.color = theColor;
          this.fruitName = theFruitName;
          this.nativeToLand = theNativeToLand;

          this.showName = function () {
              console.log("This is a " + this.fruitName);
          }
      }

<· undefined
> var f1 = new Fruit("red", "apple", "LT");
<· undefined
> f1.showName();
  This is a apple
```

Praktika (1) Duomenų patikrinimas

- Sukurkite teksto laukelį (textarea), į kurį galima suvesti tik raides ir skaičius.
- Kiti simboliai neleistini (!, @, #, ir t.t.)

Praktika (2) Duomenų patikrinimas

- Siuntos numeris sudarytas iš 13 ženklų:
  - **RN123456789LT** - registruotoji pašto korespondencijos siunta;
  - **CN123456789LT** - registruotasis siuntinys;
  - **EE123456789LT** - greitojo pašto siunta.
- Sukurkite įvesties laukelį siuntos numeriui patikrinti. Jeigu siuntos numeris įvestas teisingai, informuokite kokio tipo siunta.

Praktika (3) Asmens kodas

- Sukurkite įvesties laukelį asmens kodo patikrinimui.
- Paskutinis skaitmuo – kontrolinis skaičius.

Praktika (4) Lietuvių kalba

- Suskaičiuoti kiek žodyje, sakinyje:
  - Balsių; (*kiek kiekvienos balsės)
  - Dvigarsių (dvibalsių): ai, au, ei, ie, ui, uo
  - Skaičių;

Praktika (5) Slaptažodžio generatorius

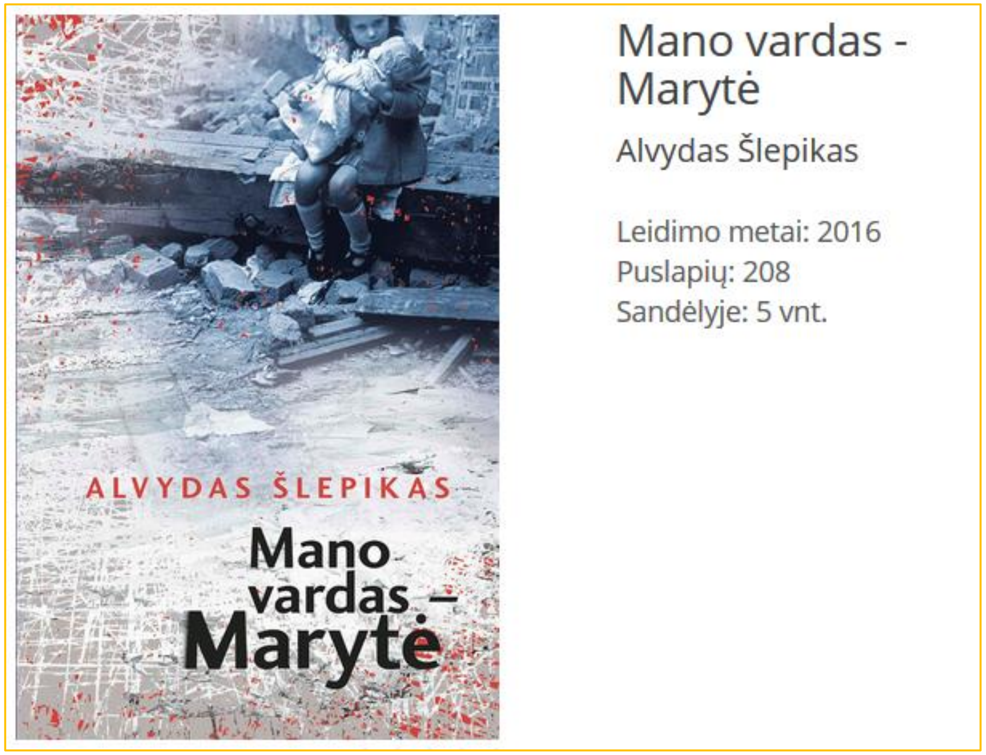
- Vartotojui leisti pasirinkti iš kokių ir kiek simbolių sugeneruoti slaptažodį:
  - 0-9;
  - !, @, #, $, ...
  - a-z, A-Z
- Papildyti įvesties lauku. Leisti įvesti žodį, iš kurio būtų generuojamas slaptažodis.

Praktika (6) E-knygynas (1)

- Knyga – objektas, kuris turi laukus: pavadinimas, autorius, leidimo metai, puslapių skaičius, liko knygų, kaina.
- Visos knygos saugomos masyve (min 3).

Praktika (6) E-knygynas (2)

- HTML dokumente pateikite knygas. Minimalus CSS.



Mano vardas - Marytė

Alvydas Šlepikas

Leidimo metai: 2016
Puslapių: 208
Sandėlyje: 5 vnt.

Praktika (6) E-knygynas (3)

- Įgyvendinkite
  - paiešką pagal pavadinimą, autorių ar kitus laukus.