13 paskaita

Code Academy

Programuok savo ateitį!

Planas

1. Modern mode, "use strict"
2. Masyvai (array)
3. Ciklai: while, for

Code Academy

Programuok savo ateitį

Modern mode

Strict mode

- 2009 m. pasirodė **ES5** ir pridėjo naują funkcionalumą bei pakeitė jau esantį. Pagal nutylėjimą - šie patobulinimai **išjungti**.

- Pridėdami “**use strict**” direktyvą failo pradžioje - mes naudojame modernią JavaScript versiją.

Strict mode

Kai naudojame “use strict”:

- Apribojame JavaScript galimybes;
- Pakeičiame sintaksę;
- Visada rodome “tylias” klaidas;
- JavaScript veikia **greičiau** (kai kuriais atvejais);
- Papildomas saugumas (**eval()** ir kt. nesaugūs veiksmai);


<code 1>

Code Academy

Programuok savo ateitį

Masyvai

Code Academy
Programuok savo ateitį!

Masyvai (array)

- **Sunumeruotas** reikšmių sąrašas/kolekcija.
- Naują masyvą galime sukurti **let myArray = []**;

- Pasiekti sąrašo elementus galime su **[] skliaustais**, pvz.: darbuotojai[0]

- Turi naudingus **metodus**, kurie gali sujungti, padalinti, surušiuoti, filtruoti..


<code 2>

Code Academy
Programuok savo ateitį

Queue (eilė)

push
prideda naują elementą į masyvo pabaigą

shift
grąžina pirmą elementą, bei pakeičia visų elementų indeksus, antras tampa pirmu.

Code Academy
Programuok savo ateitį!

Queue (eilė)

Masyvų **push**/**shift** metodai - dažnai naudojami praktikoje.

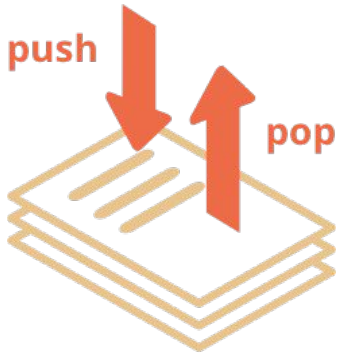
Pavyzdžiui, naujų žinučių atvaizdavimas vartotojui iš sąrašo.

Code Academy
Programuok savo ateitį!

Stack (krūva)

Yra ir kitas masyvų panaudojimo būdas - tai duomenų struktūra **stack.**

Nauji elementai visada yra pridedami arba išmetami iš masyvo pabaigos.




<code 3>

splice

Universalus metodas, kadangi gali *pridėti*, *išmesti*, bei *įterpti* masyvo elementus.

Sintaksė:
arr.splice(index[, deleteCount, elem1, ..., elemN])

Masyvo dalies kopijavimas

slice

Paprastesnis metodas nei **splice**. Jis grąžina naują masyvą, kur nukopijuoja nuo “**start**” iki “**end**” (neįtraukia “**end**”)

Sintaksė: **arr.slice(start, end)**


<code 4>

Masyvų sujungimas

concat

Sujungia dviejų masyvus elementus ir grąžina naują masyvą.

Sintaksė: **arr.concat(arg1, arg2...)**

split / join

Split padalina tekstą į naują masyvą, pagal nurodytą skirtuką (delimiter). **Join** metodas veikia atvirkščiai - sukuria tekstą iš masyvo elementų.

Sintaksė: **str.split(delim)**
arr.join(delim)

Masyvų rikiavimas

sort(fn)

Surikiuoja elementus naudodamas paduotą rikiavimo funkciją (arba rikiuoja kaip tekstą, jei nėra funkcijos)

Sintaksė: **arr.sort()**

indexOf, includes

Su **indexOf** galima surasti elemento indeksą, o jei jis nerastas mums yra grąžinamas -1. **Includes** grąžina *true* jei surandamas ieškomas elementas.

Sintaksė: **arr.indexOf(item, from)**
arr.includes(item, from)


<code 5>

Code Academy

Programuok savo ateitį

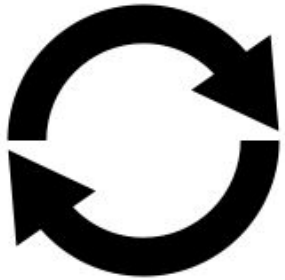
Ciklai

Code Academy
Programuok savo ateitį!

Ciklai

Dažnai tenka **kartoti** tuos pačius veiksmus. Pavyzdžiui atspausdinti prekės iš sąrašo, vieną po kitos.

Ciklai - tai **būdas kartoti** tam tikrą kodą kelis kartus.



Ciklas while

while ciklas - kol sąlyga yra **true**, kodas yra kartojamas.

```
while (condition) {
    // code
    // so-called "loop body"
}
```

Ciklas do...while

do...while ciklas - pirmiausiai įvykdo kodą, tik vėliau tikrina sąlygą, ar vykdyti dar kartą.

```
do {
  // loop body
} while (condition);
```

Ciklo nutraukimas

while ciklą sustabdyti galime 2 būdais:

- Pakeisti sąlygą į **false**
- Ciklo viduje parašyti: **break;**

Code Academy
Programuok savo ateitį!

Ciklo iteracijos užbaigimas

continue sakinys

Su juo galime atšaukti dabartinę ciklo iteraciją, bet nenutraukti viso ciklo.

Pasiekęs **continue, while** ciklas pereina prie sąlygos tikrinimo, **for** ciklas - prie **step** dalies.


<code 6>

Ciklas for

for ciklas - tai dažniausiai naudojamas ciklas.

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

begin - įvykdomas vieną kartą ciklo pradžioje

condition - tikrinama prieš kiekvieną ciklo iteraciją

step - vykdomas po kiekvienos ciklo **body** operacijos, prie sąlygos patikrinimą.

Run **begin**
→ (if **condition** → run **body** and run **step**)
→ (if **condition** → run **body** and run **step**)
→ (if **condition** → run **body** and run **step**)
→ ...


<code 7>