Web technológia

JavaScript

Dr. Hatwagner F. Miklós

Széchenyi István Egyetem, Győr

https://github.com/wajzy/GKxB_INTM049.git

2021. november 8.

Objektumok

- tulajdonság (kulcs) – érték párok (csak a `null`-nak és az `undefined`-nak nincsenek tulajdonságai a nyelvben)
- minden tulajdonság egyedi az objektumban
- a tulajdonság lehet adat vagy függvény (metódus)
- a tulajdonságot az értéktől : választja el, a párokat egymástól ,

Objektum definiálása literálként

```
1 const hg = {
2   nev: "Kovács István",
3   neptun: "a1b2c3",
4   zh: 12
5 };
```

hg kötése konstans, de ettől még a tulajdonságok értéke megváltoztatható.
Tulajdonságok elérése: `objektum.tulajdonság` formában

Objektum módosítása

```
6  console.log(hg.zh); // 12
7  hg.zh = 14; // Tulajdonságok változtathatók
8  console.log(hg.zh); // 14
9  // De const miatt az objektum nem váltható le
10 /*
11 hg = { // TypeError: invalid assignment to const 'hg'
12   nev: "Nagy Péter",
13   neptun: "1q2w3e",
14   zh: 13
15 };
16 */
```

Két kötés (referencia) ugyanarra az objektumra

```
18 // Kötések (binding), nem klasszikus változók
19 let hallgato = hg;
20 hg.zh = 15;
21 console.log(hallgato.zh); // 15
```

Tulajdonságok feltérképezése

- `in` (tartalmazás) operátor (vs. `if(objektum.tulajdonság) ...`)
- `for/in` ciklus, a tulajdonságokon történő iterálásra

Ha a tulajdonság neve kötéssel adott, a `.` operátor nem használható → objektum["tulajdonság"]

Tulajdonságok elérése

```
23 // Tulajdonság létezésének tesztelése
24 console.log("nev" in hg); // true
25 console.log("evfolyam" in hg); // false
26
27 // Milyen tulajdonságok vannak az objektumban, milyen értékkel?
28 function nyomtat(obj) {
29   for(let tul in obj) {
30     console.log(tul, ": ", obj[tul]);
31   }
32 }
```

Objektumok tartalmának másolása:

```
Object.assign(cél, forrás1, forrás2, ..., forrásN)
```

Visszatérési érték: cél

Tulajdonságok másolása

```
34 // Objektumok tartalmi másolása
35 const hg2 = {
36   nev: "Kovács Emőke",
37   zh2: 19
38 };
39
40 let egyesitett = Object.assign({}, hg, hg2);
41 nyomtat(egyesitett);
42 Object.assign(hg2, hg);
43 nyomtat(hg2);
```

Kimenet

```
nev :  Kovács Emőke
neptun :  a1b2c3
zh :  15
zh2 :  19

nev :  Kovács István
zh2 :  19
neptun :  a1b2c3
zh :  15
```

Tulajdonságok értékadással bármikor felvehetők az objektumba, és `delete` operátorral törölhetőek

Tulajdonságok hozzáadása, törlése

```
45 // Tulajdonságok utólagos hozzáadása, elvétele
46 hg.zh1 = hg.zh;
47 delete hg.zh;
48 hg.zh2 = 20;
49 nyomtat(hg);
```

Kimenet

```
nev :  Kovács István
neptun :  a1b2c3
zh1 :  15
zh2 :  20
```

Rövidített objektum definíciós szintakszis: a kötés neve lesz a tulajdonság neve is

Metódus hozzáadása

```
51 let nev = "Fekete Péter";
52 let neptun = "abcdef";
53 let zh1 = 12;
54 let zh2 = 8;
55 const hg3 = {
56   nev: nev,
57   neptun: neptun,
58   zh1: zh1,
59   zh2: zh2
60 };
61 nyomtat(hg3);
62 const hg4 = { nev, neptun, zh1, zh2 };
63 nyomtat(hg4);
```

Kimenet

```
nev :  Fekete Péter
neptun :  abcdef
zh1 :  12
zh2 :  8

nev :  Fekete Péter
neptun :  abcdef
zh1 :  12
zh2 :  8
```

Metódus: a tulajdonság értéke függvény. Az objektum többi tulajdonsága a `this`-en keresztül érhető el

Metódus hozzáadása

```
65 // Metódusok; arrow fn. nem használható,
66 // mert nincs saját kötése a this-hez
67 hg.getAlairas = function() {
68   return (this.zh1+this.zh2) >= 20;
69 }
70 console.log(hg.getAlairas()); // true
```

Tömbök

- Speciális objektumok, amelyekben a tulajdonságok nevei (kulcsok) nem negatív egész számok, de az értékek vegyesen bármilyen típusúak lehetnek
- Tömb literál létrehozása: `[elem1, elem2, ..., elemN]`
- Tömb elemszáma: `tömb.length` tulajdonság
- Elemek elérése: `[]` operátorral

Tömb létrehozása, indexelés, elemszám megállapítás

```
1  let t1 = []; // üres tömb
2  console.log(typeof t1); // object
3  let t2 = ["Alma", "Banán", "Citrom"];
4  console.log(t2[1]); // Banán
5  t2[1] = "Burgonya";
6  console.log(t2[1]); // Burgonya
7  console.log(t2.length); // 3
```

A tömb bejárására használhatóak a `for/in` (tulajdonságok/indexek) és `for/of` (értékek) ciklusok

Tömbök bejárása

```
9  function nyomtat1(tomb) {
10   for(let elem of tomb) {
11     console.log(elem);
12   }
13 }
14 nyomtat1(t2);
15
16 function nyomtat2(tomb) {
17   for(let idx in tomb) {
18     console.log(idx, ": ", tomb[idx]);
19   }
20 }
21 nyomtat2(t2);
```

Kimenet

```
Alma
Burgonya
Citrom

0 :  Alma
1 :  Burgonya
2 :  Citrom
```

Tömböt állít elő az `Object.keys()` egy objektum tulajdonságaiból

Objektum tulajdonságainak visszaadása tömbként

```
23 let tulajdonsagok = Object.keys({
24   egy: 1,
25   ketto: 2,
26   harom: 3
27 });
28 nyomtat2(tulajdonsagok);
```

Kimenet

```
0 : egy
1 : ketto
2 : harom
```

További elemek hozzáadása egy kiválasztott indexű elemhez történő hozzárendeléssel lehetséges. A tömb elemszáma a legnagyobb index alapján kerül meghatározásra, nem a tárolt elemek száma alapján!

Tömbök elemei

```
30 t2[3] = "Dió";
31 nyomtat2(t2);
32 t2[5] = "Füge";
33 console.log(t2.length); // 6
34 console.log(t2[4]); // undefined
35 nyomtat2(t2);
```

Kimenet

```
0 :  Alma
1 :  Burgonya
2 :  Citrom
3 :  Dió
6
undefined
0 :  Alma
1 :  Burgonya
2 :  Citrom
3 :  Dió
5 :  Füge
```

A literál megadásakor is jelezhetjük, hogy bizonyos indexű elemeket nem kívánunk létrehozni.

Hiányos tömbök

```
36 let t3 = ["Alma", , "Citrom", ];
37 console.log(t3.length); // 3
38 nyomtat2(t3);
39 let t4 = ["Alma", , "Citrom", undefined];
40 console.log(t4.length); // 4
41 nyomtat2(t4);
```

Kimenet

```
3
0 : Alma
2 : Citrom
4
0 : Alma
2 : Citrom
3 : undefined
```

Tömbelem törlése: delete operátorral

Tömbelem törlése

```
43 delete t4[2];
44 nyomtat2(t4);
```

Kimenet

```
0 : Alma
3 : undefined
```

Verem műveletek

- Tömb végén: `push()`/`pop()`
- Tömb elején: `unshift()`/`shift()` (vagyis egy *igazi* sort pl. a `push()`/`shift()` párossal lehetne létrehozni)

Veremműveletek

```
46  let t5 = [ 1, 2, 3 ];
47  t5.push(4);
48  nyomtat2(t5);
49  console.log(t5.pop()); // 4
50  nyomtat2(t5);
51  t5.unshift(0);
52  nyomtat2(t5);
53  console.log(t5.shift()); // 0
54  nyomtat2(t5);
```

Kimenet 1/2

```
0 : 1
1 : 2
2 : 3
3 : 4
4
0 : 1
1 : 2
2 : 3
```

Kimenet 2/2

```
0 : 0
1 : 1
2 : 2
3 : 3
0
0 : 1
1 : 2
2 : 3
```

Tömbök egyesítése: concat()

Tömbök egyesítése

```
56 let t6 = ["Alma", "Banán"];
57 let t7 = [1, 2, 3];
58 let t8 = t6.concat(t7);
59 nyomtat2(t8);
```

Kimenet

```
0 : Alma
1 : Banán
2 : 1
3 : 2
4 : 3
```

Többdimenziós tömb egydimenziós tömbök egymásba ágyazásával hozható létre

Többdimenziós tömbök

```
61 let t9 = [t6, t7];
62 function rekNyomtat(tomb, prefix="") {
63   for(let idx in tomb) {
64     if(typeof tomb[idx] == "object") {
65       rekNyomtat(tomb[idx], prefix + idx + "␣");
66     } else {
67       console.log(prefix, idx, ":␣", tomb[idx]);
68     }
69   }
70 }
71 rekNyomtat(t9);
```

Kimenet

```
0  0 :  Alma
0  1 :  Banán
1  0 :  1
1  1 :  2
1  2 :  3
```

A concat() azonban a paraméter tömb elemeit mindig külön-külön szúrja be az eredmény tömbbe! (De pl. a push() megfelelő lenne tömb beszúrására.)

Többdimenziós tömbök és concat()

```
71 rekNyomtat(t9);
72 let t10 = t9.concat(["alfa", "beta"]);
73 rekNyomtat(t10);
```

Kimenet

```
0  0 :  Alma
0  1 :  Banán
1  0 :  1
1  1 :  2
1  2 :  3
<empty string> 2 :  alfa
<empty string> 3 :  beta
```