VEZÉRLÉSI SZERKEZETEK - CIKLUSOK

- elöl tesztelős ciklus:
  - Először mindig feltétel ellenőrzés történik
    - → ha a feltétel teljesül, a ciklusmagban lévő utasítás (sorozat) végrehajtódik, majd a vezérlés ismét a feltétel ellenőrzéséhez kerül
    - → ha a feltétel nem teljesül akkor a vezérlés átadódik a ciklust követő utasításra, (azaz továbblépés , vagy kilépés a ciklusból)
  - Mivel a ciklusmagot csak akkor hajtja végre, ha a feltétel igaz, az is lehetséges, hogy a mag utasításai egyszer sem futnak le

    a feltétel mindig igaz akkor végtelen ciklust kapunk

```
i = 0;
while ( i <= 10) {
      document.write(i + "<br/>");
      i = i + 1;
}
```

VEZÉRLÉSI SZERKEZETEK - CIKLUSOK

- • Hátul tesztelős ciklus:
  - először lefut egyszer a ciklusmag → majd ellenőriz egy ciklusfeltételt.
    - → ha a feltétel még mindig teljesül, akkor a lefut ismét a ciklusmag és a vezérlés ismét a feltétel ellenőrzéséhez kerül
    - → ha a feltétel nem teljesül akkor a vezérlés átadódik a ciklust követő utasításra, (azaz továbblépés , vagy kilépés a ciklusból)

  a mag egyszer mindenképpen lefut

  Ha a feltétel mindig igaz akkor végtelen ciklust kapunk

```
i = 0;
do{
   document.write(i + "<br/>");
   i = i + 1;
}
while ( i <= 10)
```

VEZÉRLÉSI SZERKEZETEK - CIKLUSOK

- A ***for*** ciklus az előltesztelő ciklus átfogalmazása, ami addig ismétli a ciklusmagot, amíg a feltétel igaz.
- A működéséhez három paramétert(inicializálás, inkrementálás vagy dekrementálás, ciklusfeltétel) kell megadni .
  - Az inicializálásban megadhatunk egy, vagy több értékadást, amelyeket a későbbiekben ciklusváltozóként használunk.
  - Az inkrementálásban ezen változók változtatjuk, azaz nem kell egyenként növelnünk a változót.
  - A feltételben megadhatunk a ciklusváltozóra vonatkozó logikai feltételt.

inicializálás ciklusfeltétel {in|de}krementálás

```
for ( i = 0; i <= 10; i = i + 1){
  document.write(i + "<br/>");
 }
```

VEZÉRLÉSI SZERKEZETEK - CIKLUSOK

- A **for … in** *ciklust* arra használjuk, hogy végiglépkedjünk egy objektum tulajdonságain/attribútumain.
- az objektum minden attribútumát végigveszi, ezekre számszerűen hivatkozhatunk, mintha tömbelemekre hivatkoznánk.
- A feltételben meg kell adnunk az objektumot, illetve, hogy milyen változóval szeretnénk végigiterálni.

Például ha egy objektum változóinak nevét és értékét szeretnénk kiíratni:

```
<script>
    var person = {
        fname: "John",
        lname: "Doe",
        age: 25
    };
    var text = "";
    var x;
    for (x in person) {
        text += x + " : " + person[x] + "</br>";
    }
    document.getElementById("demo").innerHTML = text;
</script>
```

Példa A for ...in ciklusra

fname : John
lname : Doe
age : 25

VEZÉRLÉSÁTADÓ UTASÍTÁSOK

- A **break *[címke]*** utasítás, használtakor futás kilép a ciklusmagból, és a vezérlést átadja a ciklust követő utasításnak. Ha egy *címke* nevét írjuk a **break** után, akkor a vezérlés ahhoz a címkéhez adódik át.
- A **continue *[címke]*** utasítást, a ciklusmagban megadva abbamarad a ciklusmag végrehajtása, és a feltétel-kiértékeléshez adódik át a vezérlés, azaz a következő ciklusban folytatódik a végrehajtást. Ha megadunk egy *címkét* is az utasításnak, ekkor az adott címkénél folytatódik a végrehajtás.

```
function testBreak() {
    var i = 0;
    var x = parseInt(document.getElementById("szam").value);
    while (i < 6) {
        document.getElementById("eredmeny").innerHTML += "i: " + i + "<br>";
        if (i == 3) {
            break;
        }
        i += 1;
    }
    document.getElementById("eredmeny").innerHTML += "Eredmény:" + i*x;
}
```

Szám: 15

szamol

i: 0
i: 1
i: 2
i: 3
Eredmény:45

```
<script>
    var i = 0;
    var j = 8;

    checkiandj: while (i < 4) {
                    console.log('i: ' + i);
                    i += 1;
    checkj:     while (j > 4) {
                    console.log('j: ' + j);
                    j -= 1;
                    if ((j % 2) == 0)
                    continue checkj;
                    console.log(j + ' is odd.');
                }
    console.log('i = ' + i);
    console.log('j = ' + j);
    }
</script>
```

Vizsgáló | Konzol | H

Hálózat | CSS | JS | Bi

```
i: 0
j: 8
7 is odd.
j: 7
j: 6
5 is odd.
j: 5
i = 1
j = 4
i: 1
i = 2
j = 4
i: 2
i = 3
j = 4
i: 3
i = 4
j = 4
```

ÖSSZETETT ADATTÍPUSOK - TÖMBÖK

JavaScriptben kétféle összetett típus van: a **tömbök** és az **objektumok**.

- **Tömbök**
  - A JavaScript tömb egy 0-tól kezdve, számokkal indexelt adatszerkezet
  - az egyes tömbelemeket a [] operátorral érhetjük el.
  - nagyon dinamikus típusról → a tömb hossza tetszőlegesen változtatható, új elemeket lehet beletenni, meglévőket törölni, módosítani
  - A tömbbeli elemek tetszőleges típusúak lehetnek.
  - Tömbliterál megadásánál a [], a szögletes zárójelpárok között lehet megadni a tömb elemeit
  - A tömb hosszát a *length* tulajdonságával kérhetjük le.

Mátrixokat, többdimenziós struktúrákat tömbök tömbjeként lehet létrehozni

A tömbök tipikus feldolgozása ciklussal történik. a **for ... in** *ciklus* és **for** *ciklus* a is használható erre

```
//Üres tömb létrehozása
var uresTomb = [];

//Tömb létrehozása előre feltöltött elemekkel
var tomb = [12, 'alma', true];

//Hivatkozás a tömb elemeire
tomb[0];    // => 12;
tomb[1];    // => 'alma';
tomb[2];    // => true;

//A tömb hosszának lekérdezése
tomb.length // => 3

//Elemek módosítás
tomb[0] = 13;
tomb[0];    // => 13
```

```
//Új elem beszúrása a tömb végére
tomb[tomb.length] = 'uj';   // => tomb[3] === 'uj'

//Új elem felvétele tetszőleges indexre
tomb[100] = 'messze';
tomb.length;    // => 101

//A köztes elemek undefined értékűek
tomb[99];       // => undefined

//Elem törlése (méret nem változik)
delete tomb[1];
tomb[1]; // => undefined
tomb.length; // => 101
```

```
//Mátrix
var m = [[1, 2, 3], [4, 5, 6]];

//Vagy sokkal olvashatóbb módon:
var m = [
    [1, 2, 3],
    [4, 5, 6]
];

//Elem elérése:
m[1][2]; // => 6
```

EGYDIMENZIÓS TÖMBÖK

```
function tombs() {
    var uresTomb = [];   //Üres tömb létrehozása
    var tomb = [12, 'alma', true];      //Tömb létrehozása előre feltöltött elemekkel
    tomb_kiir(tomb);
    tomb_hossza(tomb);

    tomb[0] = 13;
    tomb_kiir2(tomb);

    //Új elem beszúrása a tömb végére
    tomb[tomb.length] = 'uj';
    tomb_kiir(tomb);
    //Új elem felvétele tetszőleges indexre - a  köztes elemek undefined értékűek
    tomb[11] = 'messze';
    tomb_kiir2(tomb);
    tomb_hossza(tomb);

    delete tomb[1];
    tomb_kiir(tomb);
    tomb_hossza(tomb);
}
 //Hivatkozás a tömb elemeire tomb[index], tömb elemeinek kiírása - az undefined elemeket nem írja ki
function tomb_kiir(tomb) {
    document.getElementById("tombok").innerHTML += "A tömb elemei:" + "<br>";
    for (var elem in tomb) {
        document.getElementById("tombok").innerHTML += "tomb[" + elem + "] = " + tomb[elem] + "<br>";
    }
    document.getElementById("tombok").innerHTML += "<hr>";
}
//Hivatkozás a tömb elemeire tomb[index], tömb elemeinek kiírása, kiírja az undefined elemek is
function tomb_kiir2(tomb) {
    document.getElementById("tombok").innerHTML += "A tömb elemei:" + "<br>";
    for (var i=0; i < tomb.length ; i++) {
        document.getElementById("tombok").innerHTML += "tomb[" + i + "] = " + tomb[i] + "<br>";
    }
    document.getElementById("tombok").innerHTML += "<hr>";
}
//A tömb hosszának lekérdezése: tomb.length
function tomb_hossza(tomb) {
    document.getElementById("tombok").innerHTML += "A tömb hossza: " + tomb.length + "<br>";
    document.getElementById("tombok").innerHTML += "<hr>";
}
```

A tömb elemei:
tomb[0] = 12
tomb[1] = alma
tomb[2] = true

A tömb hossza: 3

A tömb elemei:
tomb[0] = 13
tomb[1] = alma
tomb[2] = true

A tömb elemei:
tomb[0] = 13
tomb[1] = alma
tomb[2] = true
tomb[3] = uj

A tömb elemei:
tomb[0] = 13
tomb[1] = alma
tomb[2] = true
tomb[3] = uj
tomb[4] = undefined
tomb[5] = undefined
tomb[6] = undefined
tomb[7] = undefined
tomb[8] = undefined
tomb[9] = undefined
tomb[10] = undefined
tomb[11] = messze

A tömb hossza: 12

A tömb elemei:
tomb[0] = 13
tomb[2] = true
tomb[3] = uj
tomb[11] = messze

A tömb hossza: 12

mutat

TÖBBDIMENZIÓS TÖMBÖK

```
function tombs2D() {
    var uresTomb2D = [[], []];                  //Üres 2D tömb létrehozása
    tomb_kiir_2D(uresTomb2D);
    tomb_hossza_2D(uresTomb2D);

    var tomb2D = [[1, 2, 3], [4, 5, 6]];      //Mátrix
    tomb_kiir_2D(tomb2D);
    tomb_hossza_2D(tomb2D);

    // tömb tetszőleges elemének törlése lés felülírása
    delete tomb2D[1][1];
    tomb2D[1][2] *= 2;
    tomb_kiir_2D(tomb2D);
    tomb_hossza_2D(tomb2D);
    //Új elem beszúrása a az első sor végére végére

    tomb2D[1][tomb2D[1].length] = 'uj elem';
    tomb_kiir_2D(tomb2D);

    //Új elem felvétele tetszőleges indexre - a  köztes elemek undefined értékűek
    tomb2D[0][6] = 'sorvége';
    //tomb2D[2][0] = 'valami'; // <-- a sorok száma így nem bővíthető
    tomb_kiir_2D(tomb2D);
    tomb_hossza_2D(tomb2D);
}

function tomb_kiir_2D(tomb) {
    document.getElementById("tombok2D").innerHTML += "A tömb elemei:" + "<br>";
    for (var i = 0; i < tomb.length; i++) {
        for (var j = 0; j < tomb[i].length; j++)
            document.getElementById("tombok2D").innerHTML += "tomb[" + i + "][" + j + "] = " + tomb[i][j] + " | ";
        document.getElementById("tombok2D").innerHTML += "<br>";
    }
    document.getElementById("tombok2D").innerHTML += "<hr>";
}

function tomb_hossza_2D(tomb) {
    document.getElementById("tombok2D").innerHTML += "A tömb sorainak szám: " + tomb.length + "<br>";
    for (var i = 0; i < tomb.length; i++) {
        document.getElementById("tombok2D").innerHTML += "A tömb " + i + ". sorának hossza: " + tomb[i].length + "<br>";
    }
    document.getElementById("tombok2D").innerHTML += "<hr>";
}
```

több dimenziós tömb

A tömb elemei:

A tömb sorainak szám: 2
A tömb 0. sorának hossza: 0
A tömb 1. sorának hossza: 0

A tömb elemei:
tomb[0][0] = 1 | tomb[0][1] = 2 | tomb[0][2] = 3 |
tomb[1][0] = 4 | tomb[1][1] = 5 | tomb[1][2] = 6 |

A tömb sorainak szám: 2
A tömb 0. sorának hossza: 3
A tömb 1. sorának hossza: 3

A tömb elemei:
tomb[0][0] = 1 | tomb[0][1] = 2 | tomb[0][2] = 3 |
tomb[1][0] = 4 | tomb[1][1] = undefined | tomb[1][2] = 12 |

A tömb sorainak szám: 2
A tömb 0. sorának hossza: 3
A tömb 1. sorának hossza: 3

A tömb elemei:
tomb[0][0] = 1 | tomb[0][1] = 2 | tomb[0][2] = 3 |
tomb[1][0] = 4 | tomb[1][1] = undefined | tomb[1][2] = 12 | tomb[1][3] = uj elem |

A tömb elemei:
tomb[0][0] = 1 | tomb[0][1] = 2 | tomb[0][2] = 3 | tomb[0][3] = undefined | tomb[0][4] = undefined | tomb[0][5] = undefined | tomb[0][6] = sorvége |
tomb[1][0] = 4 | tomb[1][1] = undefined | tomb[1][2] = 12 | tomb[1][3] = uj elem |

A tömb sorainak szám: 2
A tömb 0. sorának hossza: 7
A tömb 1. sorának hossza: 4
