# Lámpák

## Fekadat

Ebben a játékban az a feladat, hogy le kell kapcsolni hét egymás melletti ablakban lévő lámpát minél kevesebb lépésben.

Ha egy ablakra rákattintunk, akkor abban az ablakban és a két szomszédos ablakban is ellenkezőjére változik a lámpa állapota: a felkapcsolt lekapcsolódik, a lekapcsolt pedig fel. A két szélső ablak is szomszédnak számít.



A játék végén meg kell kérdezni a játékos nevét, és az ablakok alatt a három legjobb eredményt meg kell jeleníteni.

## Előkészületek

1. Nyisd meg a *lampak* mappát a Visual Studio Code-ban, és nyisd meg a fájlokat a mappából!
2. Nézd át a HTML és a CSS kódot!
3. Nézd meg az oldalt a böngészőben!

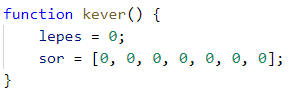
## Lámpák állapota

A lámpák állapotát egy sor nevű 7 elemű tömb segítségével modellezzük. Ha egy tömbelem 0, akkor a hozzá tartozó lámpa le van kapcsolva, ha a tömbelem 1, akkor pedig fel van kapcsolva.

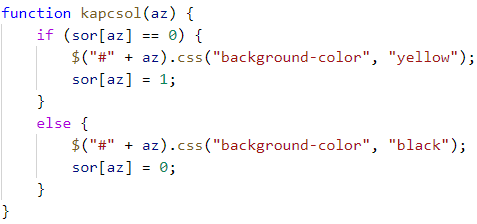
A lépések számát egy lepes nevű változóban tároljuk.

Először ezt a két változót hozzuk létre, majd egy kever nevű függvényt, amely beállítja a változók kezdőértékét, és később a lámpák kezdőállapotát is beállítja majd. Írd ezt a *lampak.js* fájlba:

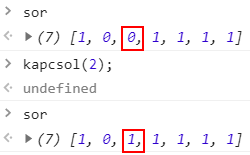




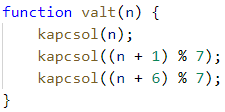
Szükségünk van egy olyan függvényre, amely egy adott lámpa állapotát elentétesre változtatja. A lámpa indexét paraméterként kapja meg. A bekapcsolt lámpák háttere sárga, a kikapcsoltaké fekete. A tömbben is jelezni kell a változást.



Próbáld ki a függvény működését a konzolról!



Kell egy olyan függvény is, amely egy lámpát és a két szomszédját együtt váltja, a széleket is figyelembe véve:

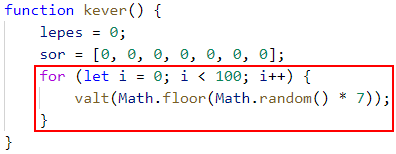


Az (n + 1) % 7 kifejezés az n-dik utáni lámpa indexe, a szélet is figyelembe véve.

Az (n + 6) % 7 kifejezés az n-dik előtti lámpa indexe, a szélet is figyelembe véve.

Próbáld ki ezt is a konzolról! Nézd meg a széleket is!

Most már el tudjuk végezni a keverést is: 100-szor választunk egy véletlen lámpát, és a szomszédaival együtt átváltjuk. Egészítsd ki a kever függvényt:

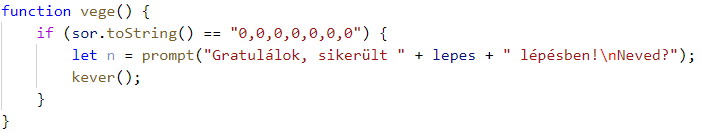


Próbáld ki! Ahányszor frissíted az oldalt, mindig más kezdőállapotnak kell lennie.

A játékos minden lépése után meg kell vizsgálni, hogy véget ért-e már a játék, vagyis minden lámpa le van-e kapcsolva. Ha vége van, gratulálni kell, és beolvasni a játékos nevét (ezt majd később használjuk fel).

A beolvasás után újra keverni kell.

Ezeket a vege nevű függvénnyel valósítjuk meg:

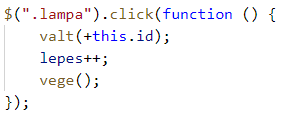


Figyeld meg, hogyan vizsgáljuk a tömb tartalmát! A toString() metódus a tömböt stringgé alakítja, és a tartalmat ugyanabban a formában adjuk meg. (Próbáld ki a sor.toString() kifejezést a konzolon!)

Két tömb összehasonlításakor mindkét tömböt stringgé alakítjuk.

A tömbök közvetlen öszehasonlítása azért nem működne, mert a változók csak a tömb címét tartalmazzák, és ez minden tömbnél más (akkor is, ha ugyanazok az elemek vannak benne).

Most már elkészítheted a lámpák eseménykezelőjét:



Az eseménykezelőben először átváltjuk azt a lámpát és szomszédait, amelyre a felhasználó rákattintott. A this ezt az elemet jelenti, az id-je megegyezik a tömbbeli indexével, csak szövegként, ezért a + jellel számmá alakítjuk.

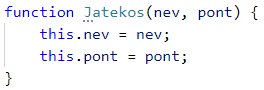
A kattintás után megnöveljük a lépésszámot, és a vege függvény segítségével ellenőrizzük a játék végét.

Próbáld ki! Most már játszható a játék, csak az eredményeket nem jeleníti meg.

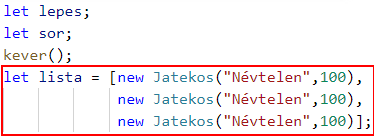
## Eredmények

Minden játékosnál a nevet és a pontszámot (lépésszámot) kell tárolni. Ezért minden játékoshoz készítünk egy objektumot, és ezeket egy tömbbe tesszük.

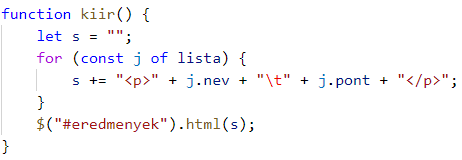
A játékos objektumok létrehozásához készítünk egy konstruktor függvényt:



A konstruktor függvény segítségével a program elején létrehozunk egy listát. Ebben Névtelen játékosok szerepelnek 100-100 lépéssel (aminél biztos tudunk kevesebbet).

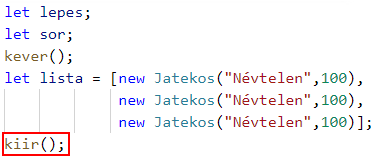


Kell egy kiir függvény is, amellyel kiíratjuk a lista tartalmát, és ezt meg is kell majd hívni a program elején és a vege metódusból is.



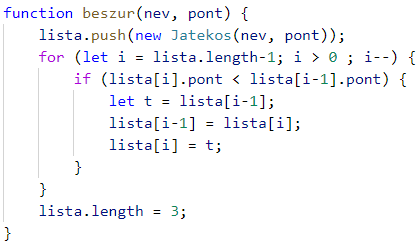
A for of ciklus úgy működik. hogy a lista minden elemét sorban egymás után beleteszi a j állandó (!) objektumba, és mindegyikkel végrehajtja a ciklust. Ebben az esetben minden listaelemhez készít egy HTML bekezdést.

Hívd meg a kiir függvényt a programból a lista létrehozása után!



Figyeld meg, hogy az oldal betöltődése után kiírja a három Névtelen játékost az eredményekhez!

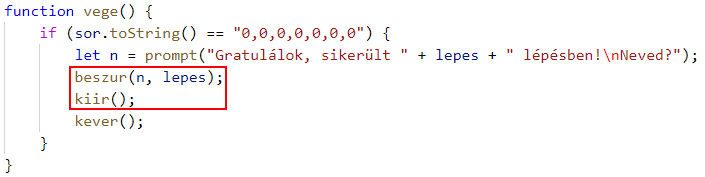
Már csak az van hátra, hogy a játék végén az új eredményt beszúrjuk a listába. Ehhez készítünk egy beszur nevű függvényt, amely paraméterként kapja meg a játékos nevét és pontszámát (vagyis lépésszámát):



Itt először létrehozunk egy új játékos objektumot, és a lista négéhez fűzzük.

Utána Végignézzük visszafelé a lista tömb elemeit a harmadiktól az elsőig. Ha valamelyik elem kisebb az előtte lévőnél, akkor megcseréljük őket. Így az elemek növekvő sorrendbe kerülnek, és a három legjobb lesz a lista elején.

Végül a lista hosszát 3-ra állítjuk, és így levágjuk a negyedik elemet.

A játék végén, a vege függvényben be kell szúrni a játékos eredményét a listába, és újra ki kell írni a listát: 

Ezzel elkészült a program. Teszteld!