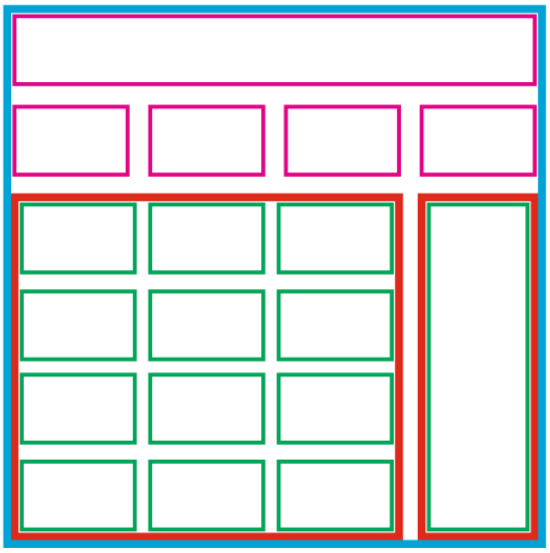
Calculator projekt

Számos módja lehet a calculator projekt megoldásának. Gondoltam összeírom vázlatosan, hogy milyen lépésekkel haladtam.

1. HTML és CSS - a számológép megjelenésének összeállítása.
Szerintem ezt a feladatot GRID-del sokkal egyszerűbb összerakni, ettől függetlenül erőltettem a FLEX-es megjelenítést, hisz majd később ezt fogjuk sokat használni.
A számológép törzsének megépítéséhez az alábbi struktúrát alkalmaztam:



fő konténer

kijelző

összeadás kivonás szorzás osztás

konténer 1
7 8 9
4 5 6
1 2 3
0 . C
/konténer 1

konténer 2
egyenlőségjel
/konténer 2

/fő konténer

2. Javascript - a gombok eseménykezelőinek elkészítése.
A szám és műveleti jel gombokra tettem közös css class-t, amelyre megírtam egy eseménykezelőt, amely elsőként annyit tett, hogy kiírta a gomb textContent-jét a konzolra.

3. Javascript - megnyomott gombok tartalmának kiíratása a "kijelzőre".
A két eseménykezelőm eddig a konzolra dolgozott. Ebben a fázisban annyit tettem, hogy a kijelző textContent-jét mindig bővítem a leütött gomb textContent-jével.

4. Javascript – műveletlánc szeparálása
Ez eddig oké, de mi lesz akkor, amikor majd kiolvasom ezt a műveletláncot? Hogy bontom fel elemeire?
Erre az az ötletem támadt, hogy a számok és műveletek közé jó lenne egy elválasztó karakter, ami mentén szét tudom majd választani őket. Erre a szóköz volt a leggusztusosabb megoldás, hisz ez a kijelzőn is jól mutat.
A kérdés azt volt, hogyan kerül be ez a szóköz? Az az ötletem támadt, hogy ha megnyomok egy műveleti jel gombot, akkor annak tartalma elé és mögé kerüljön egy-egy szóköz. Ehhez viszont el kellett különítenem a szám és a műveleti jel gombokat egymástól. Erre egy-egy külön css class-t vezettem be. Ennek okán az eseménykezelőből is kettő kellett, egyik a számokra, amit meghagytam az eredetiből. Egyet pedig a műveleti jelre, ami lényegében ugyanaz, mint a másik, kiegészítve az elő és utó szóközzel.

5. Javascript - C gomb eseménykezelője és függvénye.
Ez egy könnyebb fázis. A C gomb is külön CSS class-t kapott. Erre készítettem egy eseménykezelőt 'click' eseményre, ami a meghívott függvényben annyit tesz, hogy a kijelző értékét törli, azaz .textContent = '';

6. Javascript - Az = gomb eseménykezelője és függvénye.
Nálam az = gomb is egyedi CSS class-t kapott. Erre írtam egy ugyanolyan eseménykezelőt, mint az eddigiek, ami egy olyan függvényt hív meg, amely majd feldolgozza a kijelzőre írt műveletláncot és a kijelzőre írja az eredményt. De elsőként csak annyit tett, hogy kiemelte és egy string-ként eltérolta egy változóban.

7. Javascript - A string felbontása tömbbé.
Mivel már okosban előkészítettem a kijelzőm tartalmát a szóközös szeparálással, ezért split-tel a szóközök mentél szét tudtam a stringet bombázni tömbbé. A kapott tömbömben szám, jel, szám, jel sorrendben váltakoztak az elemek, amelyek még mind string típusúak.

8. Javascript - A tömb szétválogatása számokká és műveleti jelekké
A következő lépésben létrehoztam két üres tömböt. Egyet a számoknak, egyet a jeleknek.
A meglévő tömböt iterálva például for ciklussal, vagy forEach-csel - kinek melyik megy jobban - a páros indexűeket a szám tömbbe, a páratlanok a művelet tömbbe push-oltam.
A szám tömbbe push-olásnál elvégeztem a számmá parse-olást is. Én itt parseFloat-ot használtam. Nem foglalkoztam azzal, hogy biztonságos számot, azaz nem óriás értéket kap az input.

9. Javascript - A műveletek elvégzése és az eredmény kiírása
Ehhez egyébként Gáll Gergely is adott segítséget, de nem is túl "bonyi" a feladat, viszont meg kellett látni benne az logikát.
Ha nem nyomkodtunk be hülyeséget a kijelzőre, akkor ugyebár szám - jel - szám - jel - szám ... jel - szám sorrendben követték egymást az elemek. Ami fontos, hogy egyel több számunk van, mint jelünk.
Hogy érdemes haladni? Én így csináltam.
Létrehoztam egy eredmény változót 0 értékkel, majd gyorsan hozzá is adtam a szám tömböm 0. indexű elemét.

Legyen egy egyszerű példánk: 3 + 2 * 5 – 10
Ebből ez a két tömb született:

szám = [3, 2, 5, 10]
jel = ['+', '*', '-']

Látjuk, hogy a szám négy elemű, a jel csak három.
Tehát az eredményt máris növeltem a szám tömb 0. elemével, azaz 3-mal. eredmény = 3
Ezt követően bejártam a jel tömböt.

i = 0: '+' - kell tehát egy összeadás. Mit mivel? Az eredményt a következő számmal, tehát eredmény és jel[0] és szám[1], azaz eredmény és jel[i] és szám[i+1]

A példára lefordítva:

A jel 0. indexe esetén eredmény = eredmény + 2 = 3 + 2 = 5
A jel 1. indexe esetén eredmény = eredmény * 5 = 5 * 5 = 25
A jel 2. indexe esetén eredmény = eredmény - 10 = 25 - 10 = 15
Remélem, érthető volt!

A kérdés még az lehet, hogy ez oké, de a műveletek elvégzését miként választom ki és végeztettem el? A legegyszerűbb, ha if - else if-ekkel felveszitek a műveleteket!
pl.:

```
function müveletek(operator, szam1, szam2) {
    if (operator === '+') eredmény = szam1 + szam2;
    else if (operator === '-') eredmény = szam1 - szam2;
    else if (operator === 'x') eredmény = szam1 * szam2;
    else if (operator === '÷') eredmény = szam1 / szam2;
}
```

Én más módon csináltam. Egy objetumban vettem fel paraméterként az összes műveletet, hisz függvény lehet opbjektum paramétere is.

```
const operatorsObject = {
   '+': (num1, num2) => (num1 + num2),
   '-': (num1, num2) => (num1 - num2),
   'x': (num1, num2) => (num1 * num2),
   '÷': (num1, num2) => (num1 / num2)
 };
```