HTML+JAVASCRIPT „kódkölcsönző” programozási gyorstalpaló kezdőknek/laikusoknak

(HTML+JAVASCRIPT programming – case study for beginners based on the principle of the code-integration)

Pitlik László, Pitlik Mátyás, MY-X team

Kivonat: Az alábbi esettanulmány egyszerre kívánja szolgálni a duális képzések megalapozását IT-területen, ill. különösen a nem IT-képzések IT-orientált részleteinek, vagyis a laikus/kezdő programozóknak a megszólítását. A cél annak demonstrálása, miként lehet teljesen nulláról eljutni egy valós céges/intézményi igényként megfogalmazódó, valóban repetitív, azaz automatizálást elváró feladat megoldásáig. Az autodidakta programozni tanulást az Interneten nagyon sok helyen és formában rendelkezésre álló megoldások támogatják, melyek különösen HTML és JAVASCRIPT kódok előállítása kapcsán szinte kiáltanak a „kódrészlet-kölcsönzés” stratégiájának és operatív trükkjeinek alkalmazása után. Az Olvasó bevezetésre kerül egy olyan világba, ahol, ha jól kérdez valaki, akkor szinte mindig kap megfelelő válaszokat. Az autodidakta megoldás egyetlen egy kockázata tehát a jól kérdezés mibenlétében érhető tetten. Lehet-e az, hogy valóban laikusok jól kérdezzenek? Elég-e egy ilyen esettanulmány az átlag-laikusnak, hogy a jól kérdezés kompetenciáját elsajátítsa, ahol egy ilyen leírás távoktatási anyagként kellene, hogy támogassa az életen át tartó tanulás erőtereit bármely kiképzendő bármely életszakaszában?

Kulcsszavak: just-in-time-tudásmenedzsment

Abstract: The paper (this case study) tries to serve two parallel aims: both the preparation of the dual educations on the field of the IT and the support processes of non-IT-related beginners needing progress in programming. The case study presents a process where an exactly defined task should be finalized from the first character of the source code to a functioning solution. This quasi learning process is an autodidact experiment where already existing codes should be integrated and finetuned. Fortunately, puzzle-elements and test environments can be easily identified online for HTML and/or JAVASCRIPT source-codes. The Readers will be introduced into a world where good questions lead to good puzzle-elements and finetuning strategies. How it is possible to find the proper keywords? It is relatively easy: practice makes perfect. Relevant question is whether beginners are capable of questioning in an appropriate way? Is this paper enough to support the beginners working in an autodidact way on the field of the lifelong learning?

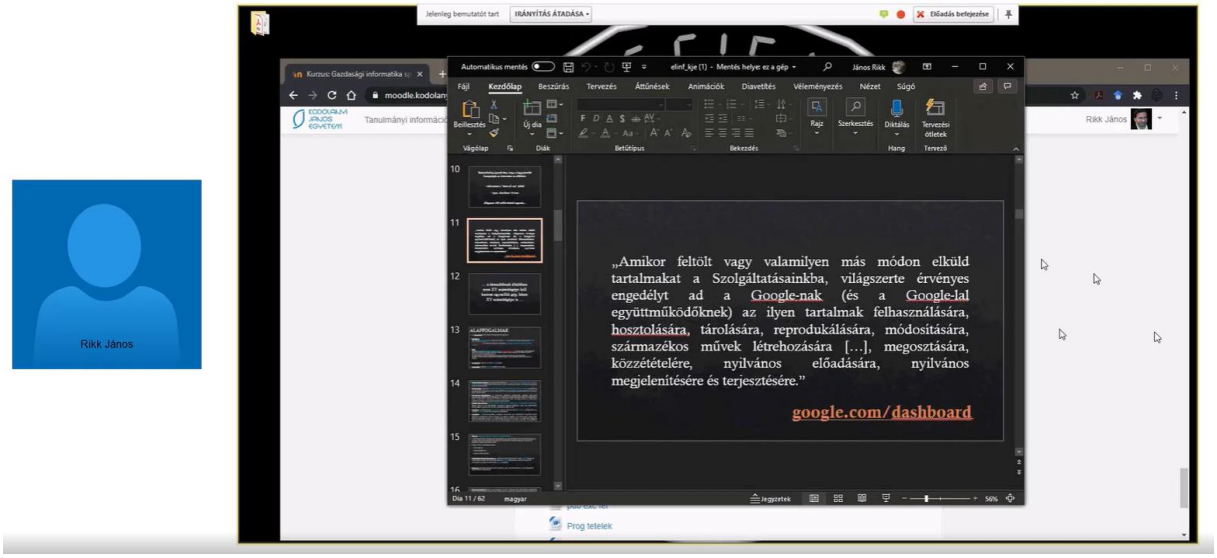
Keywords: just-in-time-knowledge management

Bevezetés

A COVID-helyzet kapcsán lehetséges elvárás, hogy egy adott intézmény/cég dolgozói/tanulói naponta jelezzék, hogy a kritikusnak ítélt jelenségek kapcsán vannak-e említésre méltó tapasztalatik. Egy ilyen napi jelentés keletkezzen pl. egy Google-kérdőív segítségével, melyből a lekérhető CSV-kivonat legyen egy vesszővel elválasztott állomány. Ezen állomány tagolt értelmezése vezessen a következő xls-állományhoz: https://miau.my-x.hu/bprof/anonim_version.xlsx, ahol a riportok feladata a név szerint érintettek kilistázása.

Megjegyzendő, hogy az itt és most valós alapokból értelmezhetetlenné átkódolt, de strukturálisan valós példa éles adatokkal jobb lenne, ha sosem kerülne a Google szervereire, de az kétségtelen, hogy saját szerver fenntartása és egy egyébként ingyenesen elérhető kérdőívezési szolgáltatás saját

kialakítása vélhetően kellően drága, kockázatos és bizonytalan kihívás ahhoz, hogy egy cég/intézmény nem IT-specialista vezetői ne a Google mellett döntsenek, főleg időkényszerek mellett. Ahhoz, hogy a Google-szerű külső adatkezelés teljesen kizárható legyen, törvényi tiltást kellene kimondani.



Rikk János előadásának (2020.09.04.) egy részletét kiemelve belátható, a Google-nak átadott tartalom és a GDPR elvek között masszív szakadék vélelmezhető.

A fentebb hivatkozott XLS-állomány az Excel kimutatás-varázslási lehetőségeire támaszkodva (ill. pl. a DARABTELI() függvény manuális alkalmazásai révén adott céges/intézményi munkatársat képessé tehet arra, hogy naponta, manuálisan feldolgozza a neki megküldött CSV-állományt, ahol a ideális esetben a feldolgozásért felelős dolgozó maga kérdezi le a Google adatbázisát, hogy minél kevesebb személy férjen hozzá egészségügyi(-jellegű? – azaz védett) adatvagyonhoz, ahol persze a feltöltő (kérdőívet kitöltő dolgozók/diákok) elvileg akaratlagos hozzájárulásukat adták a feltöltéssel, hogy ezen adatok a járványügyi intézkedések támogatására (de pl. nem a katalógus-kényszer új megoldásaként) kiértékelésre kerüljenek. S pl. minden nemleges válasz quasi azonnal (ill. egy adott türelmi/biztonsági időintervallum letelését követően, mely legyen jelenleg a 14 napos önkéntes/kötelező karantén ideje) mindenhonnan törlésre kerülnek. Így a Google szerveriről is - (kérdés: ez hogyan garantálható? – ill. garantálható-e egyáltalán? – vagy legalább egy felülírás esetén csak valamiféle Google-archívumból lehetne csak a múltat előbányászni nem akármilyen jogosultságok birtokában?)…

Miért HTML+JAVASCRIPT?

A kivonatban és a címben a laikusokra történő utalás, s az autodidakta tanulás természetesen az első szakkifejezések felmerülésekor megbicsaklani látszik. S valóban el kell ismerni: egy teljesen laikustól nem várható el semmilyen szakszó ismerete. Tőle a Megrendelőkkel szembeni követelmények teljesítése várható el, s ezek ismeretében lehet abban reménykedni, hogy egy Internetes keresés a Megrendelői igények kapcsán már feldob olyan dokumentumokat, melyekben ott az utalás a megoldás felé vezető kulcsszavakra. Azt is meg kell jegyezni, hogy rossz az az oktatási rendszer, mely a munkavállalási kor előtti állami képzésekkel nem tudja elérni, hogy az átlagpolgár fejében nyomot hagyjanak olyan kulcsszavak, mint a HTML, és/vagy a JAVASCRIPT. Amennyiben az előbbi utalás az Internetes keresésre nem vezetne eredményre, akkor a laikus Megrendelő – s egyben önfejlesztésben motiváltan érdekelt személy még mindig fordulhat pontszerűen egy (hozzáképest) IT-szakértőhöz (mentorhoz, tutorhoz, tanácsadóhoz – vagy egyszerűen csak a szomszédhoz, a lépcsőházban egyfolytában IT-eszközökkel szaladgáló ifjú titánhoz, a gyermek informatika tanárához, az e-

Magyarország-pont dolgozójához, stb.), akik egyikétől elvárható, hogy a szükséges kulcsszavakat meg tudja adni az érdeklődő számára.

Elsőként nézzük tehát a Megrendelői elvárás szómágiáját a mire is lenne szükség kérdés kapcsán:

- Kellene egy olyan alkalmazás/szoftver/program/MEGOLDÁS,
- Mely egy táblázatnyi (kérdőíves) adatból
- Automatikusan kilistázza
- Azokat a sorokat
- Melyek esetében a válaszadó valamire IGEN-t mondott.

A fenti tartalmi elvárások mellett megfogalmazódhatnak egyéb elvárások is a laikus fejében, vagy a laikus számára a problémát/feladatot kiadó főnök fejében (aki, mint alternatív IT-tudásforrás, már sok kulcsszót hozzá tehet a kiindulási állapot finomhangolásához) pl.

- A megoldáshoz nem legyen szükség szerverre, mert nincs és ennek léte rendszergazda létét várja el, aki vagy elérhető vagy nem, vagy gyorsan/hatásosan reagál vagy nem, vagyis a szerver egy olyan függőség, mely itt és most inkább kockázat, mint előny…
- A megoldáshoz ne kelljen semmit telepíteni, ami nincs egy laikus számítógépén (pl. ne kelljen programozást megalapozó környezetet kialakítani)…
- A megoldás offline (is) fusson, de az sem baj, ha online is hozzáférhető…
- A megoldás lehet csúnya, ha célszerű…
- A megoldás lehet nagy is, ha nem irracionálisan nagy…
- A megoldáshoz ne kelljen semmilyen licence (vö. MS Office)…
- A megoldás tényleg automatizált legyen, vagyis a napi CSV birtokában bárki képes legyen az elvárt eredmény kinyerésére…
- A megoldás nem kell, hogy bolond-biztos legyen, mert pl. két egymástól független egyszerű megoldás, mely azonos eredményt ad, lehet elég a napi kockázatmenedzsmenthez – szemben egy nagyon részletgazdagon kimunkált hibakezelést is nyújtó szoftverrel…
- A megoldás legyen olcsó és gyorsan előállítható… (ne kelljen tanfolyamra járni ennek kifejlesztéséhez még egy laikusnak sem), stb.

Speciális feladat (vagyis az esettanulmány kereteit meghaladó kihívás) ezen a ponton annak bizonyítása, hogy a fenti elvárások alapján az Internetes keresések képesek elvezetni a HTML-JAVASCRIPT kulcsszavakhoz anélkül, hogy a kereső a megoldást előre tudná!

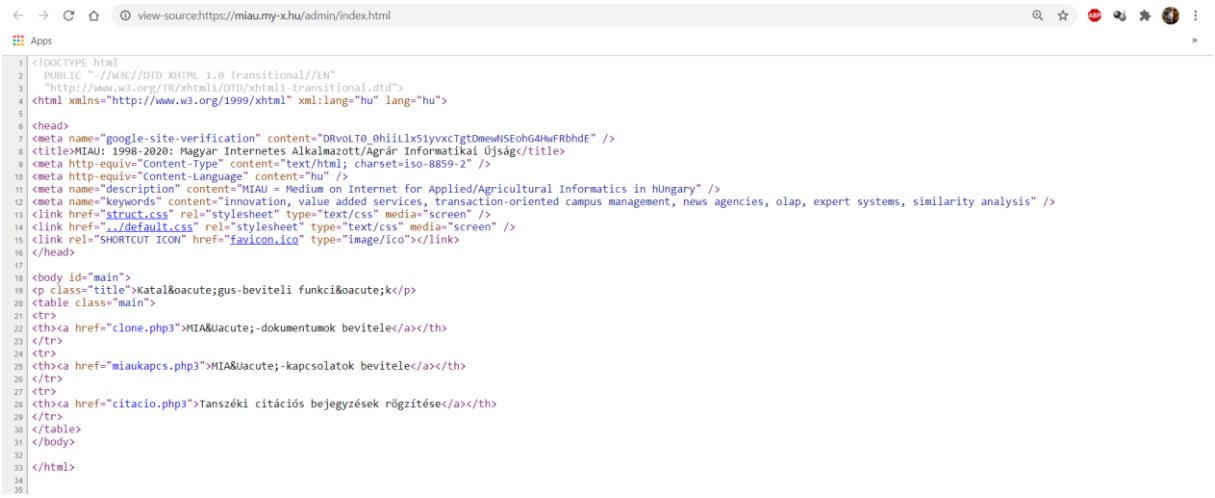
A fentiek alapján a HTML+JAVASCRIPT megoldás racionálisnak tűnik, mert

- HTML-t lehet konvertálni pl. szövegszerkesztőkben, táblázatkezelőkben előállított dokumentumok kapcsán, s egy egyszerű HTML-oldal alig igényel „programozást”.
- Az Internetes HMTL oldalak CTRL+U, ill. F12 esetén a böngészőkben részletgazdagon megismerhetők.
- A JAVASCRIPT futtatása nem igényel keretrendszert, elég egy böngésző (F12 - konzol), ill. valamilyen online támogatás a jegyzettömbben is megírható kódok eredményének ellenőrzéséhez.
- A HTML-írása sem igényel keretrendszert: itt is elég egy jegyzettömb.
- A HTML-oldalakból indított JAVASCRIPT kódok is láttathatók a böngészőkben (CTRL+U, F12).

- A JAVASCRIPT-kód offline fut, de a HTML-oldalakba integrálva úm. web-ről is letölthető (s persze ekkor is „csak” offline fut).
- A JAVASCRIPT-ek írása az Interneten fellelhető minták alapján megfeleltethető a „kódkölcsönzés” stratégiájának.
- A HTML-ek írása az Interneten fellelhető minták alapján szintén megfeleltethető a „kódkölcsönzés” stratégiájának.

Első lépések: a HTML-burok a nyers adatok JAVASCRIPT általi átvételének visszaellenőrzése érdekében

Tegyük fel, hogy a megoldást kereső laikus személy rendszeres (jelszavak is birtokló, azaz komoly) felhasználója több online szolgáltatásnak, pl. a https://miau.my-x.hu szervernek – így ismer olyan szerver-oldalakat, melyek egyszerűek (nincs bennük animáció, kép, komplex struktúra – vö fejléc, lábléc, oldalmenü, stb.): pl. https://miau.my-x.hu/admin/index.html



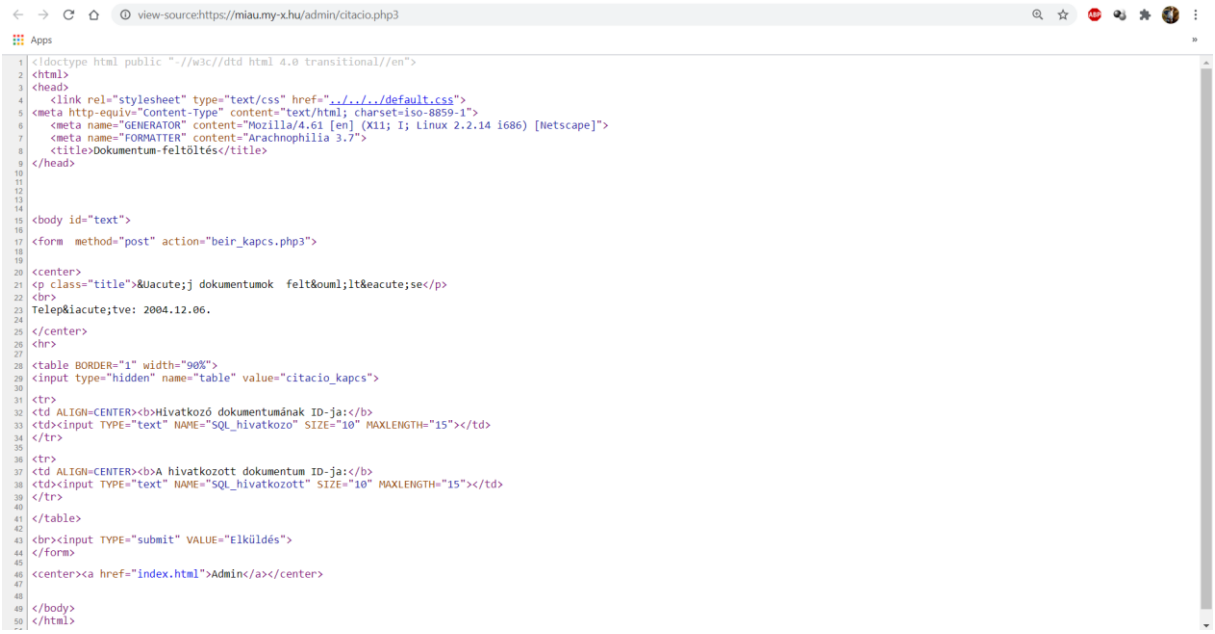
Az első HTML-kód alapján a laikusnak a következőket illik észre vennie, vagy ilyen esettanulmányok alapján tudatosítania:

- Minden betű/szó/szám/jel alapvetően nem véletlenül van a kódban.
- A különböző színű részletek mást és mást jelentenek.
- Az azonos színű részletek kb. ugyanazt jelentik/ugyanolyan módon értelmezendők.
- Minden programkód egy fájl, egy valamilyen kiterjesztéssel bíró állomány: jelen esetben itt és most az index.html-ről van szó. Vagyis a jegyzettömbből majd egy covid_riport.html állományt kell elmenteni.
- A laikus minden részletre (pl. DOCTYPE, meta name, body, stb.) rákereshet a weben, ha többet kíván ezekről tudni. De keresés nélkül is végezhet kísérleteket a saját maga által írt kódok hatását illetően. Például elhagyható-e a szürke réteg? Kell-e a „meta”, a „title”? Mi történik az oldallal, ha nincs „link”? Kell-e egyáltalán bármi a „head” blokkból? A személyes kísérletek eredményeként a laikus szakirodalmi támogatás nélkül is rá fog jönni arra, hogy a <html></html> tag-eken túl lényegében semmi más nem kell.
- Kitapasztalható az is, hogy szükség van-e a barna részletekre a minimális funkcionalitáshoz vagy sem.
- A laikus látja, milyen szintaktika vezet adott szöveg megjelenítéséhez, ill. érzékeli, hogy az ékezetes betűk gondot okozhatnak – így elsőre akár lemondhat ezek használatáról (vö. a megoldás lehet csúnya, csak hatásos legyen)…

- A „table” kapcsán szembesül azzal, hogy quasi minden megjelenítést el lehet képzelni egy táblázat részeként, ami a covid-riportok kapcsán kifejezetten hasznos felismerés.
- Amennyiben valamilyen kódrészlet nem a megfelelő helyre kerül a kísérletezés során (pl. „tr”, „td”, akkor ennek vizuális eredménye azonnal láthatóvá válik. Illetve, ha valami ugyan kódrészletként létezik, de adott helyen nem hat, az is tetten érhető lesz…
- …

A laikus következő kérdése az illene, hogy legyen, miként fog tudni egy CSV-állományt átadni a HTML-oldalon keresztül a számítógépnek annak érdekében, hogy ennek ún. átvételét a bevitt adatok táblázatként való megjelenítésén keresztül visszaigazolja a program.

Természetesen nem csak egy minta-HTML-t illik megnézni, így előbb-utóbb fellelhető lesz olyan oldal is, ahol a felhasználói oldalról valamiféle INPUT vihető be, nem csak kattintható linkek vezethetnek akcióhoz: pl.



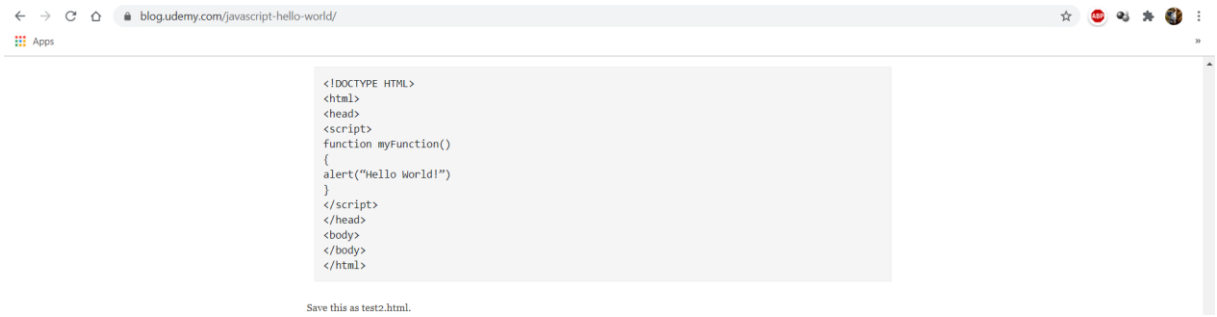
Amennyiben a laikus számára az INPUT szócska bármi oknál fogva világossá válik, akkor az adatokat bekérő HML-oldalak mögötti (CTRL+U) nézetben már nem lesz nehéz felfedezni a megfelelő tag-eket (pl. <input type=”text” …>.

S ezzel a HMTL kapcsán az elvárások és az igények szinte már egymásra is találtak, vagyis a laikus el kell, hogy jusson annak felismeréséig, hogy a következő lépés már a JAVASCRIPT hatókörébe kell, hogy tartozzon.

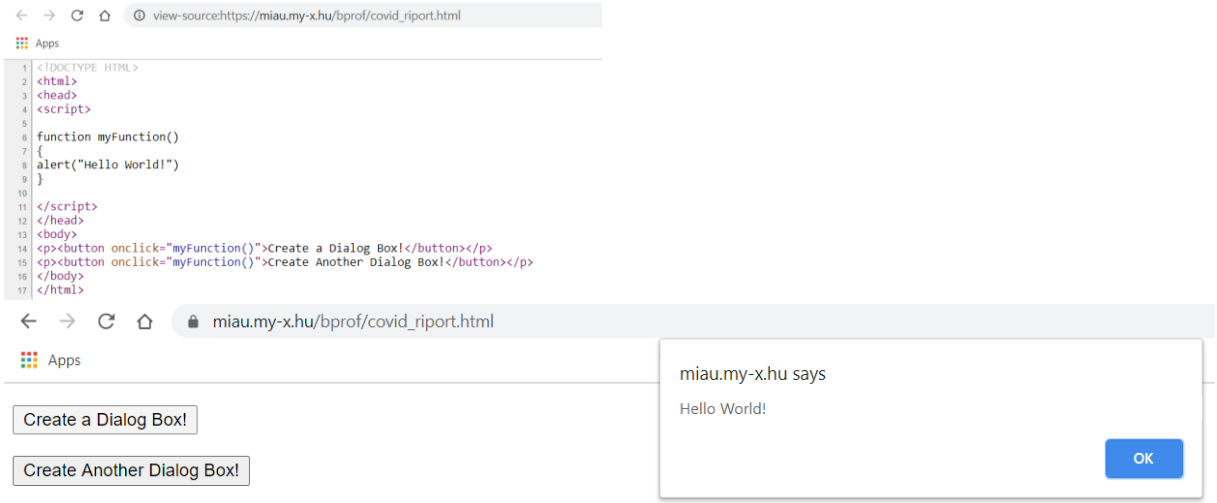
A HTML és a JAVASCRIPT rétegek kapcsolatának megteremtéséhez még természetesen statikus tartalmak JAVASCRIPT-en keresztüli megjelenítését célszerű megismerni.

Aki a programozással kacérkodik, annak előbb-utóbb szembe kell, hogy jöjjön a „hello-world”-re való utalás. Így erre és a JAVASCRIPT-re keresve az első találat a következő: https://blog.udemy.com/javascript-hello-world/

Ugyan az oldal angolul van, de nem kell hozzá nagy fantázia, s az oldalt a böngésző amúgy is képes lehet automatikusan lefordítani, hogy beazonosításra kerüljön az első HTML-be ágyazott JAVASCRIPT-kód:



Vagyis a HTML már ismerni vélt logikája (vö. nyitó és záró tag-ek egymásba ágyazva) tökéletesen alkalmas egy <script>…</script> blokk befogadására.



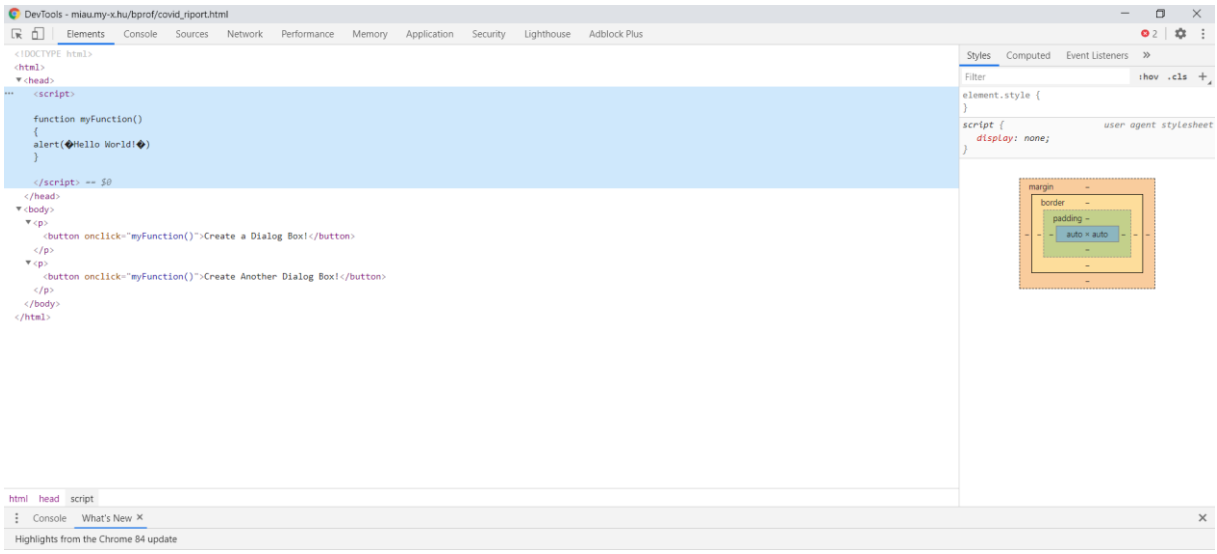
Amennyiben az angol nyelvű leírásból még integrálásra kerül a „button”-ra való utalás is ÉS az idézőjelet nem csak egyszerűen másoljuk a web-oldal látható karaktereként, hanem lecseréljük az éppen a billentyűzetünkön érvényes idézőjelekre, nos akkor máris működni fog az első HTML+JAVASCRIPT kísérletünk.

Fontos tehát: az idézőjelek (s minden más speciális jel) legyen szabványos!

```
function myFunction()
{
alert(“Hello World!”)
}
```

Nem megfelelő idézőjelek kapcsán nem fog történni semmi. A háttérben azonban a böngésző (jelen esetben egy Chrome) számos támogatást nyújthat a probléma beazonosításához: pl.

CTRL+U (vagyis a forráskód-nézet) máris utal arra, hogy nem idézőjelként érti a webről átvett és a saját html-állományunkba másolt idézőjelszerű jeleket a futtató erőtér.



Az F12 segítségével és a nyilakon keresztül a tartalmak részletes megmutatásával hasonló hibajelzést láthatunk.

Az előző „elements” mellett a „console” által adott jelzés alapján azt tudjuk meg, hogy a JAVASCRIPT myFunction() része, melyet a „button” egérkattintásra (onclick) akar aktiválni, nem értelmezhető a rendszer keretében. Ebből még nem tudjuk, mi is lett elrontva, de már tudjuk, hogy nem várható el semmilyen működés az egérkattintás hatására.

Itt érdemes egy pillanatra megállni és összefoglalni az eddigieket, különös tekintettel arra, vajon milyen esélye van eddig eljutni a laikusnak autodidakta módon a HTML és a JAVASCRIPT kulcsszavak ismerete után:

- Önállóan csak a HMTL-szálon a HTML logikája átlagos fantáziával és némi kísérletező kedv mellett gyorsan átlátható (vö. trial and error – hiszen nagy kárt nem lehet okozni rossz HTML-kódokkal).
- A JAVASCRIPT kapcsán az első lépésnek remélhető hello-world-jelenség nem csak a JAVASCRIPT-ről magáról, hanem ennek HTML-be integrálásáról is azonnal érdemi felvilágosítást ad.

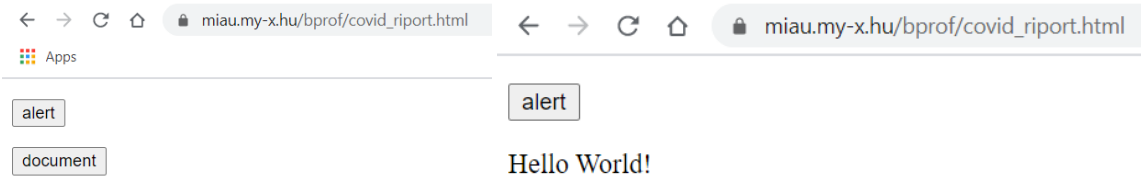
A következő logikus elvárása lehet egy laikusnak, hogy az „alert” speciális megjelenítési rétege helyett valahogy visszataláljon a HTML-hez, mint megjelenítési kerethez.

Forrás: https://www.w3schools.com/js/exercise.asp?filename=exercise_functions2

/var/www/miau/data/bprof/covid_riport.html - pitlik@my-x.hu - Szerkesztő - WinSCP

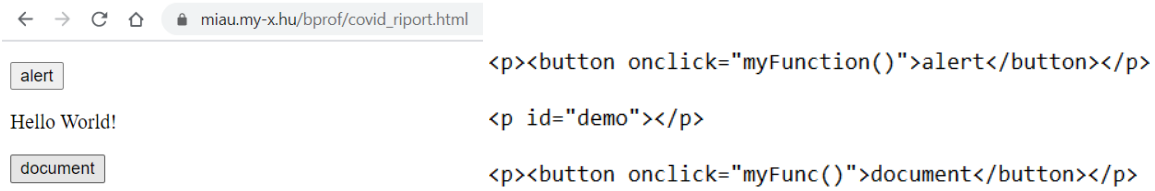
Kódolás Szín

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function myFunction()
{
alert("Hello World!")
}
function myFunc()
{
document.getElementById("demo").innerHTML = "Hello World!";
}
</script>
</head>
<body>

<p><button onclick="myFunction()">alert</button></p>
<p id="demo"><button onclick="myFunc()">document</button></p>


</body>
</html>
```

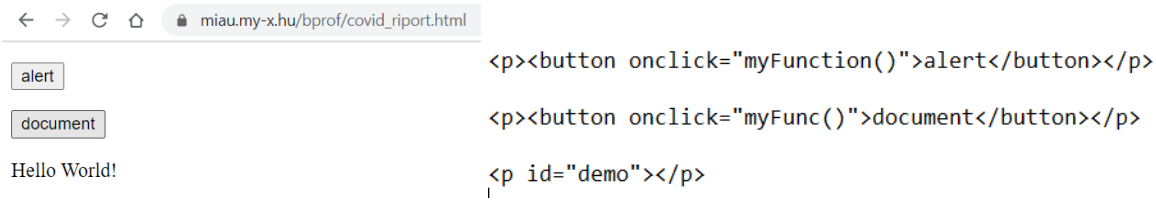


Amennyiben a webes keresés során, a „-alert” kifejezést is használjuk (pl. javascript hello world function html -alert), akkor nagy eséllyel futunk bele olyan kódmintába, mely már az elvárások szerint a képernyőre ír ki (vö. document.getElementById("demo").innerHTML = "Hello World!";).

A nyomógomb helyett, helyén való megjelenítés mellett létezik a nyomógomb megtartása melletti kiírás lehetősége is: pl.



Sőt: a kiírás helye is befolyásolható: pl.



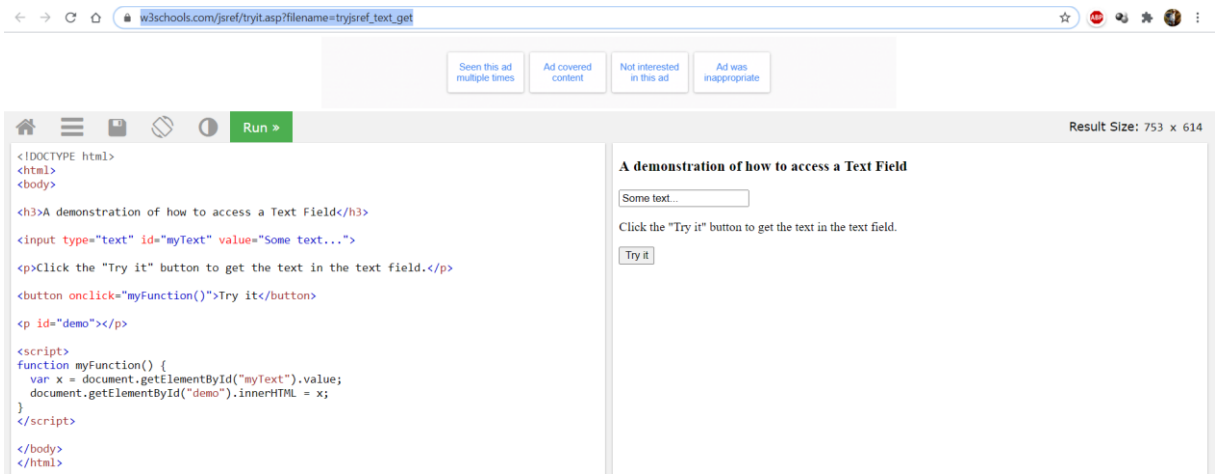
Az utóbbi lehetőség-felvillantások már a klasszikus vegytiszta programozás-tanítás trükk-fázisát idézik, ami itt és most nem cél. A just-in-time-tudásmenedzsment (tanulás/oktatás-stratégia) célja újra és újra kiemelve nem más, mint egy adott feladat érdekében ott és akkor a még hiányzó tudáselemek minél gyorsabb fellelése – s a nem belső motivációból fakadó (= nem valaki által megfizetendő bérért végzett) raktárra tanulás minimalizálása.

A következő kérdés a „Hello-World” változóként való értelmezni tudása, vagyis egy tetszőleges, legfőképpen a felhasználó által HTML-inputként átadandó szöveg megjelenítése.

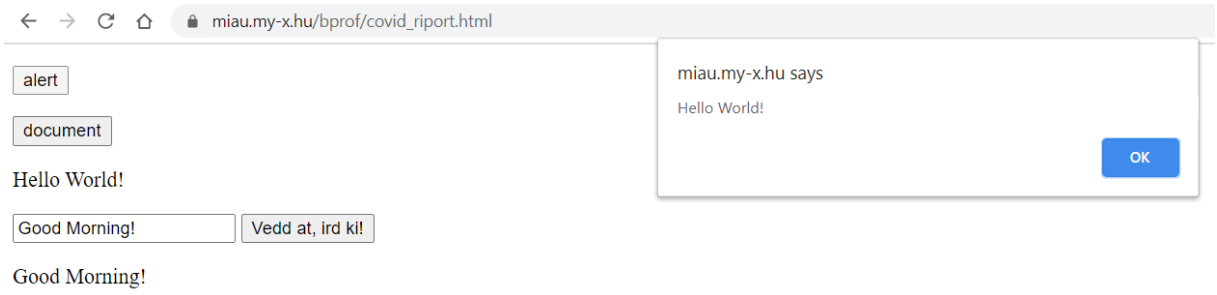
Adott tartalom megadása a programkódban lehet statikus és lehet dinamikus. Ha dinamikus, akkor ezt valamilyen ún. változóhoz kell kötni, mely értéke (tartalma) feltételek függvényében változik. A feltétel lehet pl. az idő múlása is (vagyis, ha a JAVASCRIPT figyeli a rendszer belső óráját, akkor a „Hello” helyett tudhatna a napszaknak megfelelően is köszönni – persze csak akkor, ha a napszaknak megfelelő statikus köszöntések be vannak tárolva a kódban valahol.

Mivel már tudjuk, hogy a HTML kapcsán az INPUT kulcsszóra kell keresni, az új keresési kifejezés legyen: pl. javascript html input

A megfelelő találatok triviális léteznek: pl. https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_text_get



Ha integráljuk a fentiek alapján a saját inputmezőnket a már meglévő covid_riport.html állományba, akkor a „Hello-World” mellett képessé válunk saját köszöntés kiírására is: pl.



S a kód maga így néz ki (vö. következő kép), ahol elvárás, hogy rájöjjünk, érdemes azonos nevekkel alapvetően nem kísérletezni, vagyis pl. a myFunction, demo neveket indexálni (pl. demo2). Az is érdemes immár kimondani, hogy a JAVASCRIPT kódrészletek a megfelelő tag-ek között bárhol elhelyezhetők a HTML oldal forráskódjában. Az egyes HTML-paraméterek, mint pl. a „…” az input-mező kiindulási értékeként nem kötelező, elhagyható, de éppenséggel lehet hasznos, hogy egy default üzenet az input-mezőben is elhelyezhető.

A változó az alábbi kódban az „x”, mely értékét a „myText” input-mezőtől veszi át és tovább is adja a HTML-oldal megfelelő részletének (<p id=demo2>.

/var/www/miau/data/bprof/covid_riport.html - pitlik@my-x.hu - Szerkesztő - WinSCP

Kódolás Szín

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function myFunction()
{
alert("Hello World!")
}
function myFunc()
{
document.getElementById("demo").innerHTML = "Hello World!";
}
</script>
</head>
<body>

<p><button onclick="myFunction()">alert</button></p>

<p><button onclick="myFunc()">document</button></p>

<p id="demo"></p>

<input type="text" id="myText" value="...">

<button onclick="myFunction2()">Vedd at, ird ki!</button>

<p id="demo2"></p>

<script>
function myFunction2() {
  var x = document.getElementById("myText").value;
  document.getElementById("demo2").innerHTML = x;
}
</script>

</body>
</html>
```

Itt az ideje immár a próbakód helyett egy olyat összerakni a mások által az Interneten már felkínált LEGO-elemek felhasználásával, mely új kód egy CSV-t képes átvenni és kiírni a HTML-oldalra – jelezve, hogy a CSV tartalma már volt egy JAVASCRIPT-változó tartalma, azaz tovább feldolgozásra rendelkezésre áll: pl.

/var/www/miau/data/bprof/csv_demo - pitlik@my-x.hu - Szerkesztő - WinSCP

Kódolás Szín

```
ido,nev,email,tagozat,helyszin,neptunkod,kerdes1,kerdes2,kerdes3,kerdes4
1,xy,xy@a.hu,nappali,valahol,123456,Nem,Nem,Nem,Nem
2,qw,qw@b.hu,levelezo,mashol,234567,Nem,Igen,Nem,Nem
```

A CSV és ennek jelképes tartalma…

/var/www/miau/data/bprof/covid_riport_small.html - pitlik@my-x.hu - Szerkesztő - WinSCP

Kódolás ▾ Szín ▾

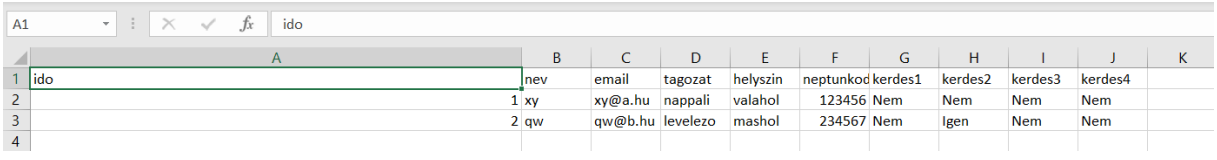
```
<!DOCTYPE HTML>
<html>
<head>

<textarea id="myText" rows="5" cols= 100" value="torold ki ezt az uzenetet es masold ide a CSV tartalmat"></textarea>

<button onclick="myFunction2()">Vedd at, ird ki ellenorzeskeppen!</button>

<p id="demo2"></p>

<script>
function myFunction2() {
  var x = document.getElementById("myText").value;
  document.getElementById("demo2").innerHTML = x;
}
</script>

</body>
</html>
```

A lényegében csak a „Hello-World”-től mentesített kód – bevezetve a „textarea” fogalmát, de a korábbi input-type is működik természetesen. A textarea annak

A HTML és JAVASCRIPT kódrészletek eredménye, ahol a bemeneti sortörések az output-ból látszólag teljesen hiányoznak.

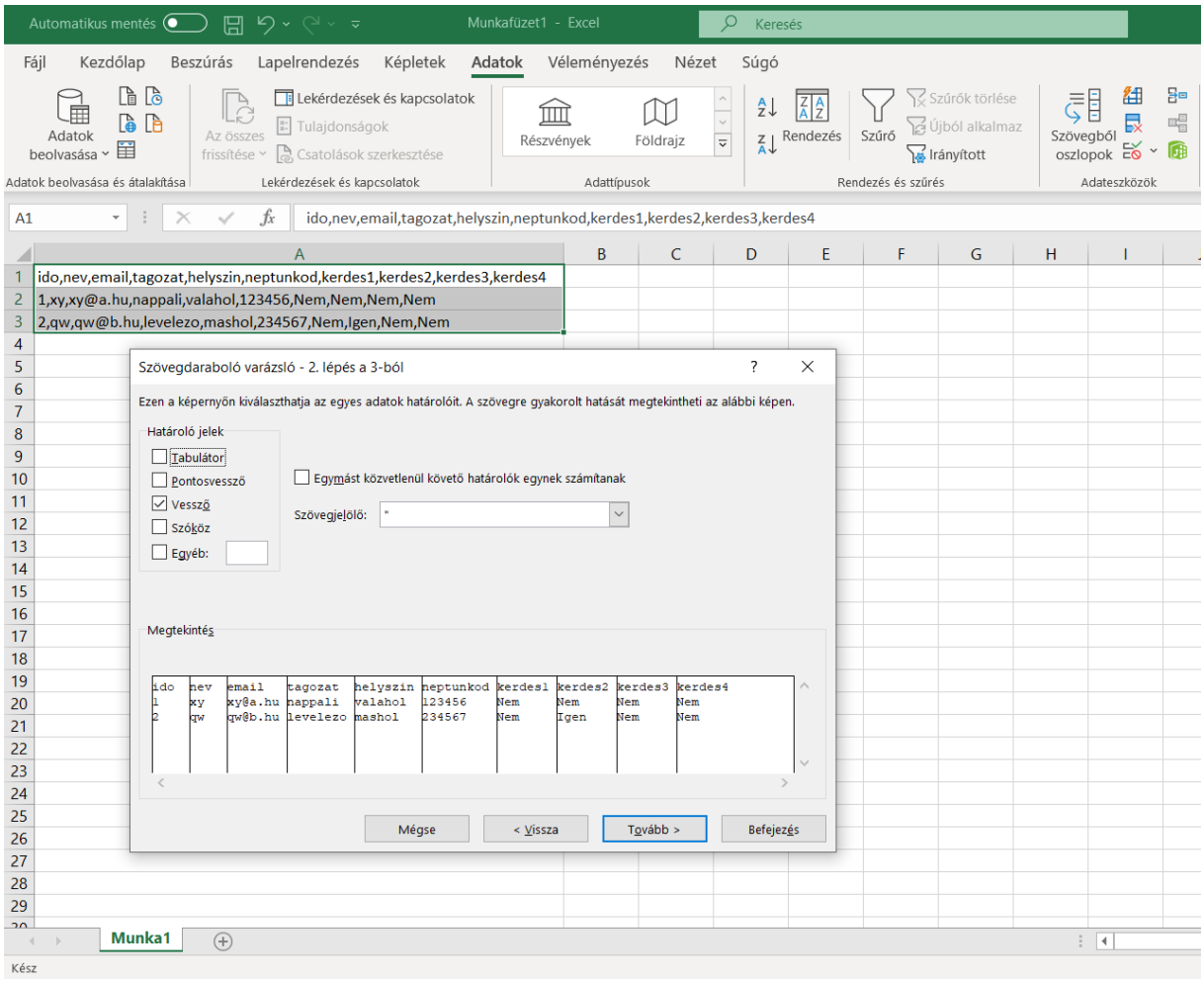
Azt is észre lehet venni, hogy a value=”torold ki…” kódrészletnek semmi hatása nincs a covid_riport_small.html megnyitásakor.

A következő feladat adja magát: műveleteket kell tudni végezni az „x” változó tartalmán annak érdekében, hogy minden egyes adathoz egyenként címezhetően hozzáférjünk, mint amikor a CSV az Excel-be kerül, s látjuk, hogy az egyetlen igen a H3-as jelű cellában van miután a „szövegből-oszlop” menüpont segítségével, a vesszőt tagolójelként beállítva megkaptuk a több-oszlopos nézetét az eredetileg 1-oszlopos CSV-másolatnak.



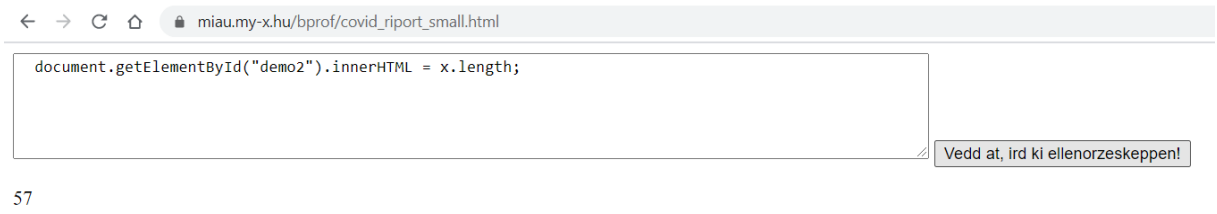
A1 | ido

	A	B	C	D	E	F	G	H	I	J	K
1	ido	nev	email	tagozat	helyszin	neptunkod	kerdes1	kerdes2	kerdes3	kerdes4	
2	1	xy	xy@a.hu	nappali	valahol	123456	Nem	Nem	Nem	Nem	
3	2	qw	qw@b.hu	levelezo	mashol	234567	Nem	Igen	Nem	Nem	
4											

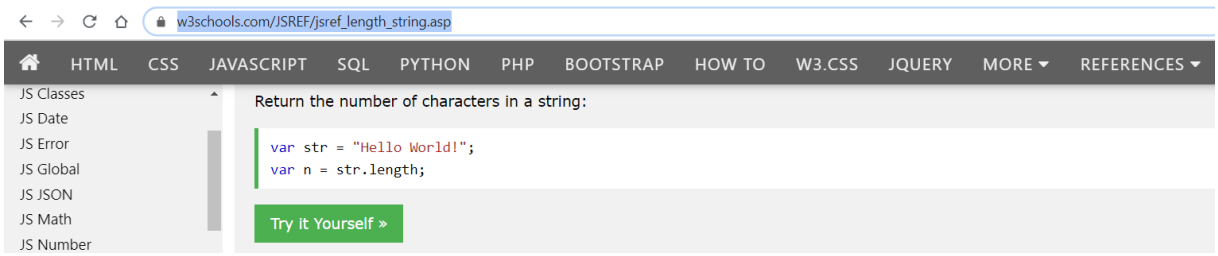


A JAVASCRIPT „x”-ként jelölt változójában tehát egy szöveg van, melyben a más esetekben sortörést eredményező jel a HTML-nézetben szóközként hat pl. a kerdes4 és az 1 karakterek között, ill. a Nem és a 2 karakterek között (s a feltöltéstől függően esetleg a HTML-kiírás legvégén is, ha a CSV bemásolásakor a kurzor nem az utolsó Nem-válasz mögött lett volna közvetlenül).

Hogy az „x”-változó tartalmával tudunk-e műveletet végezni, ahhoz elsőként érdemes valami egyszerűbbet elvárni: pl. megtudjuk-e mondani, milyen hosszú (azaz hány karakter) az „x”-változóban tárolt jelek mennyisége?



A megfelelő megoldást bárki azonnal megtalálhatja, ha rákeres pl. a javascript text length szavakra (vö. https://www.w3schools.com/JSREF/jsref_length_string.asp)



Ki kell emelni immár azt is, hogy a weben számos segítség van a JAVASCRIPT kódrészletek kipróbálására: pl. https://www.w3schools.com/JSREF/tryit.asp?filename=tryjsref_length_string

Közben azt is megérezhetik az erre fogékonyak, hogy a hossz meghatározását végző műveletre való utalás a változóhoz kapcsolódik egy fajta tulajdonságként, nem úgy, mint pl. az Excelben: =HOSSZ(„Hello World”)

S azt is fel lehet fedezni a figyelmesebbek által, hogy a változót nem csak úgy használjuk, hanem illik úm. definiálni: vö. „var str=”Hello World”;

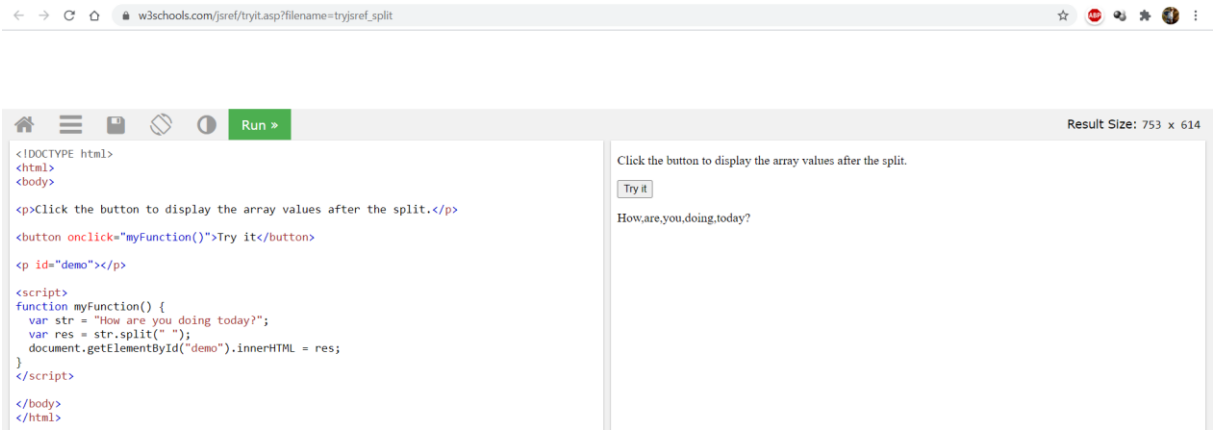
Ezek a felismerések majd akkor válnak érdemi fontosságúvá, amikor egyes, eddig gondot nem okozó, de nem ideális megoldások ideáltól való eltérése lesz egyes téves működések/nem működések oka.

Ilyen gyanúként máris felmerülhet, hogy a CSV mérete vajon lehet-e quasi korlátlan? Illetve mit is kell tenni, hogy a CSV változó mérete ne okozzon majd a jövőben gondot a valameddig jól működő megoldás felhasználójának egyszer csak?!

Most, hogy tudjuk, hogy a CSV (egyelőre demo-méretű) tartalma átadható a HTML oldal segítségével a JAVASCRIPT-nek és manipulálható a JAVASCRIPT által úgy, hogy az eredmény ismét a HTML része lesz, immár akarhatunk célirányosan is beavatkozni az eddigi folyamatba a CSV tartalmának feldolgozása érdekében: ez a célirányosság pedig nem lehet más, mint a sorok számának felismerése, vagyis annak a kontroll-értéknek a képzése, mely alapján majd az egyes kérdésekre adott egyes válaszlehetőségek darabszámának meghatározása után elvárjuk, hogy ezen részösszegek összege kiadja az összes sor értékét.

A megoldáshoz ismét csak kereséssel kell és lehet közelebb jutni: pl. javascript delimiter string

https://www.w3schools.com/jsref/jsref_split.asp

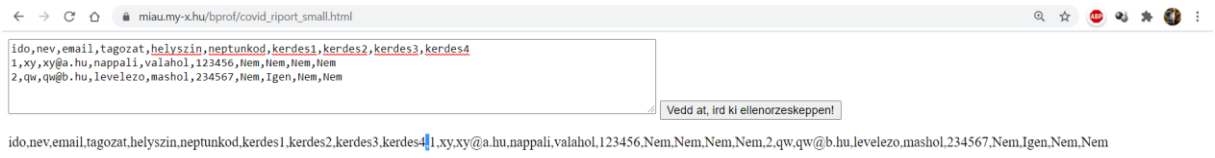


A split() segítségével megadhatunk a zárójelben egy delimitert (tagolójelet), mely esetünkben először a sorvégjel, ill. soron belül a vessző lesz.

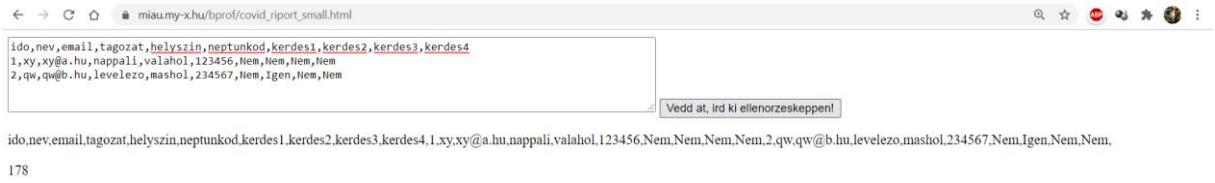
A sorvégjel (end of line) mibenlétét kereséssel szintén meg lehet határozni: vö. „\n”

Így a kód immár nem más, mint:

```
<!DOCTYPE HTML>
<html>
<head>

<textarea id="myText" rows="5" cols= 100" value="torold ki ezt az uzenetet
es masold ide a CSV tartalmat"></textarea>

<button onclick="myFunction2()">Vedd at, ird ki ellenorzeskeppen!</button>

<p id="demo2"></p>

<script>
function myFunction2() {
  var x = document.getElementById("myText").value;
  var res = x.split("\n");
  document.getElementById("demo2").innerHTML = res;
}
</script>

</body>
</html>
```



Az eredmény szemre nem látványos, de a kis két kijelölés jelzi, hogy az új „res” változó már minden delimiter pozícióban vesszőt használ. S ha valaki előre tudja, hogy a bejövő CSV-ben a mezők száma = ido,nev,email,tagozat,helyszin,neptunkod,kerdes1,kerdes2,kerdes3,kerdes4 = 10, akkor (ugyan nem lesz a kódja teljesen univerzális, de egy GDPR-érintettségű adatkezelésnél ez nem is cél), vagyis minden 10. vessző a sortörés – kivéve a legutolsó vessző hiányát. De ez is feloldható immár: pl.
„`res = res+",";`



A 178-as érték a „res” egy vesszővel megtoldott hossza, vagyis a kódunk most éppen így néz ki:

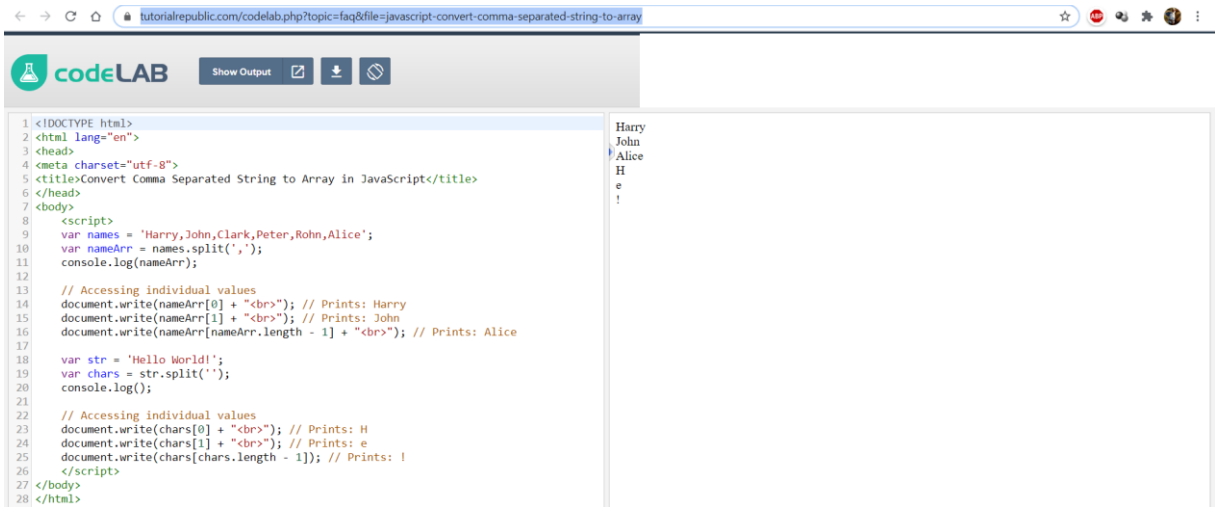
```
<!DOCTYPE HTML>
<html>
<head>

<textarea id="myText" rows="5" cols= 100" value="torold ki ezt az uzenetet
es masold ide a CSV tartalmat"></textarea>

<button onclick="myFunction2()">Vedd at, ird ki ellenorzeskeppen!</button>

<p id="demo2"></p>

<script>
function myFunction2() {
  var x = document.getElementById("myText").value;
  var res = x.split("\n");
  res = res+",";
  document.getElementById("demo2").innerHTML = res;
  document.getElementById("demo3").innerHTML = res.length;
}
</script>

<p id="demo3"></p>

</body>
</html>
```

Mint látható, a demo=3 azonosító lehetővé tette, hogy a hossz önálló kiírásként jelenjen meg a HTML oldal alján.

Most már csak az egyes mezőnevek és/vagy adatpozíciók címzése a feladat:

Ha a split() vagy az explode() kapcsán keresgélünk még, akkor olyan tartalmak villannak fel, melyek több részkérdésre is választ sejtetnek: pl. https://www.tutorialrepublic.com/codelab.php?topic=faq&file=javascript-convert-comma-separated-string-to-array

Olyan részkérdésekre, mint pl. lehet-e a res változón belül az egyes pozíciókat beazonosítani? Lehet-e ezeket a vektor/mátrix-pozíciókat önállóan kiíratni?



Az új kód először létrehozza a res2 változót, melyen belül az egyes adatpozíció (mezőnevek és adatok) a 10-mezős alapadat (alapstruktúra) kapcsán tételesen értelmezhetők, vagyis a 0., a 10. és a 20. res2()-érték magának az idő mezőnek a két értéke (1 és 2).

```
<!DOCTYPE HTML>
<html>
<head>

<textarea id="myText" rows="5" cols= 100" value="torold ki ezt az uzenetet
es masold ide a CSV tartalmat"></textarea>

<button onclick="myFunction2()">Vedd at, ird ki ellenorzeskeppen!</button>

<p id="demo2"></p>

<script>
function myFunction2() {
  var x = document.getElementById("myText").value;
  var res = x.split("\n");
  res = res+",";
  document.getElementById("demo2").innerHTML = res;
  document.getElementById("demo3").innerHTML = res.length;

  var res2 = res.split(",");
  document.write(res2[0] + "<br>");
  document.write(res2[10] + "<br>");
  document.write(res2[20] + "<br>");

}
</script>

<p id="demo3"></p>

</body>
</html>
```

Ráadásul a kiíratás logikája is megváltozott: eltűnt a bemeneti képernyő és egy üres HTML-oldalra kerülnek fel immár az első mező neve és ennek értékei.

Vagyis mostantól semmi nem állja útját annak, hogy a bemeneti CSV az elküldés után üres HTML oldalon ismét táblázatos formában jelenjen meg a JAVASCRIPT hatására. Azonban ehhez – hogy ne kelljen egyesével minden mezőnevet és hozzátartozó értéket pozícionálni, szükség lesz a JAVASCRIPT-en belül az első ciklus életre keltésére. (A split egy rejtett ismétlődésként már sok műveletet elvégzett, de ez nem általunk volt vezérelve közvetlenül.)

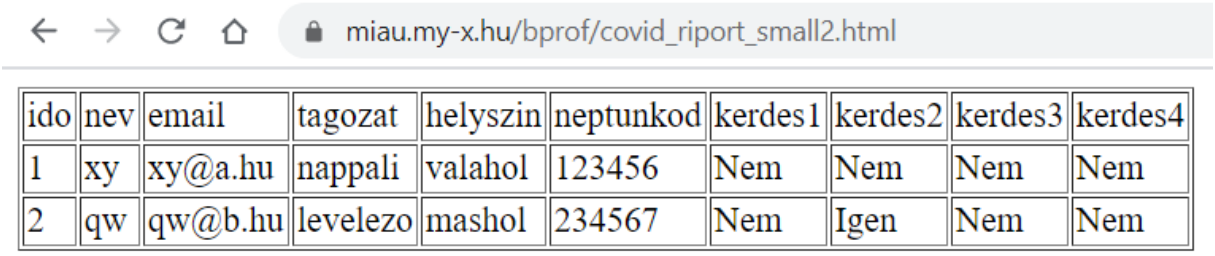
Lassacskán itt az idő azt kiemelni, hogy klasszikus programozó vélhetően soha nem mondott volna le a „\n” kapcsán arról, hogy azonnal 3-sor és 10-oszlop nézetben (tömbként) lássa és címezhesse az összes CSV-adatot, vagyis az első éles adatsor időértéke a tömb(2,1) lenne, vagyis a második sor első oszlopa (ha a számozást nem nullától indítjuk most éppen didaktikai okok miatt). Ettől még a kód működni fog azonban, legfeljebb nagyobb adatmennyiségek esetén lassabb lesz, vagy fellépnek a fejlesztés során olyan problémák, melyek más megközelítés esetén soha nem merültek volna fel.

A ciklus szintaktikáját és logikáját megérteni nem lehet nagy kihívás önmagában: pl. https://viktortaylor.wordpress.com/javascript/javascript-for-ciklus/

```
for (i=0; i <=20; ++i) { document.write(i, ''); }
illetve
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>Szorzótábla</title>
<style>
#szorzotabla
{
border : solid 1px green ;
box-shadow : 4px 4px 4px #6A6 ;
border-radius : 8px ;
background-color : #3A3 ;
}
#szorzotabla tr td
{
width : 24px ;
text-align : right ;
padding-right : 4px ;
line-height : 20px ;
font-family : Arial ;
font-size : 12px ;
border-radius : 8px ;
background-color : #FFF ;
}
</style>
</head>
<body>
<s.c.r.i.p.t type="text/javascript">
 document.write( "<table align=center border='1' id='szorzotabla'>" );
 for( i=1; i<=10; i++ )
 {
```

```
 document.write( "<tr>" );
 for( j=1; j<=10; j++ )
 {
 if( i==1 || j==1) document.write( "<td width='5%' >" + i*j + "</td>" );
 else document.write( "<td title='" +i+ "*" +j+ "'>" + i*j + "</td>" );
 }
 document.write( "</tr>" );
 }
 document.write( "</table>" );
</s.c.r.i.p.t>
</body>
</html>
```

Integrálva ezen tudáselemet a saját kódunkba az alábbi eredményt és megoldást (kódot) kapjuk, ahol a kódból immár minden felesleges rész kivezetésre került (vö. value, innerHTML, stb.)



ido	nev	email	tagozat	helyszin	neptunkod	kerdes1	kerdes2	kerdes3	kerdes4
1	xy	xy@a.hu	nappali	valahol	123456	Nem	Nem	Nem	Nem
2	qw	qw@b.hu	levelezo	mashol	234567	Nem	Igen	Nem	Nem

Akármilyen úton módon is sikerült eljutni a 3*10 pozíció táblázatként való értelmezéséhez, legkésőbb most egy klasszikus tömb immár a táblázati értékekkel a kiírással párhuzamosan feltölthetővé válik.

```
<!DOCTYPE HTML>
<html>
<head>
<textarea id="myText" rows="5" cols= 100"></textarea>
<button onclick="myFunction2()">Vedd at, ird ki ellenorzeskeppen!</button>
<script>
function myFunction2() {
  var x = document.getElementById("myText").value;
  var res = x.split("\n");
  res = res+",";
  var res2 = res.split(",");
document.write( "<table border='1'>" );
document.write( "<tbody>" );
for ( j=0; j<3; ++j )
{
document.write( "<tr>" );
for ( i=0; i<10; ++i)   { document.write( "<td>"+res2[10*j+i]+"</td>" );  }
document.write( "</tr>" );
}
document.write( "</tbody>" );
document.write( "</table>" );
}
</script>
</body>
</html>
```

A kiírás kapcsán a jelenleg 0-29, azaz 30 elemű listát mátrixként értelmezni már csak annyit jelent, hogy egy res3(j,i) hozzárendelésnek is léteznie kell, ahol res3 csak és kizárólag azért jön létre, hogy eleget tegyünk a 2-dimenziós címzés elvárásának. Itt kell megjegyezni, hogy a jelenleg még statikus sorszám (=3) és oszlopszám (=10) értékek is ideális esetben a CSV átadása után azonnal keletkeznek, hogy a CSV méretei már azelőtt meglegyenek, hogy ezzel bármi egyéb művelet elvégzésre kerülne.

A feladat tehát létrehozni egy új 2-dimenziós tömböt:

https://stackoverflow.com/questions/16512182/how-to-create-empty-2d-array-in-javascript

```
var res3 = [[],[]];
…
for ( i=0; i<10; ++i) { res3[j,i]=res2[10*j+i]; document.write(
"<td>"+res3[[j],[i]]+"</td>" ); }
```

ahol ez a megoldás ekvivalens a fentivel:

```
for ( i=0; i<10; ++i) { res3[[j],[i]]=res2[10*j+i]; document.write(
"<td>"+res3[[j],[i]]+"</td>" ); }
```

Következésképpen már csak az a lépés van hátra, ahol végre bevezetésre kerülhet a HA-logika, vagyis annak vizsgálata, mikor mondott adott kérdésre valaki igent?

A HA-szintaktikát ismét csak kereséssel kell/lehet feltárni… De mielőtt áttérne a laikus a saját naiv logikáját megkoronázó záró lépésre, ahol a fejlécet mindenkor ki akarja íratni, s alá csak azokat a sorokat, melyekben adott kérdés esetén „Igen” szerepel, tegyük fel, hogy egy szakértő a laikus válla felett belepillantott a kódba, s nem tudta megállni, hogy egy-két megjegyzést ne tegyen. Ilyeneket például, hogy:

- Miért égette be a laikus a 3-as és a 10-es értéket a kódba? (Erre a válasz még egyszerű: azért, mert ezen paraméterek, vagyis a sorok számának és az oszlopok számának dinamikussá tétele a kódírás következő lépése lesz…)
    - Arra a szakértői kérdésre, hogy ezt hogyan képzeli el a laikus, a válasz csak annyi volt, hogy a split utáni állapotok elvileg meg kellene, hogy adják a tömbök méreteit valahogy, de erre majd még rá kell keresni…
    - A laikus még azt is feltételezte, hogy ha volt értelme a „length”-nek egy szöveg esetén, akkor valami hasonló megoldás a tömbökre (pl. res, res2) is lehetne éppen adott…
- Miért kellett a következő kódrészlet a megoldásba: `res = res+",";` ? (A válasz erre a kérdésre már csak látszólag egyszerű: azért, hogy minden értékes adat mögött meglegyen a hozzátartozó vessző, hiszen ez egy fajta szabályosságot garantál, ami hátha fontos lesz még…)
    - Arra a szakértői kérdésre, hogy mi is a res a vessző hozzáadása után, a laikus válasza határozottan az volt, hogy szöveg, melyben immár minden érdemi tartalmi egység után van egy vessző.
    - Arra a szakértői kérdésre, hogy mi lett a res-változóval, mint tömbbel, a laikus nem tudott érdemben reagálni.
- Miből gondolta a laikus, hogy a `res = res+",";` egyáltalán működőképes művelet lesz, hiszen a „split” eredményeként a res lényegében már egy tömb? (A válasz: mivel a korábbi HTML-oldalba való kiíratási tapasztalatok azt mutatták, hogy a „tömb” teljes tartalma csak a változó nevére hivatkozással - vagyis mindennemű zárójel, index, stb. nélkül - szövegként jelenik meg, sőt, a sorok közötti sorvégjel is vesszővé alakul át, így vélhetően ez a tartalom létező, tovább

feldolgozható, így egy szöveget kiegészíteni egy szöveggel - egy vesszővel – miért is ne lehetne, s persze végül ez sikerült is… 😊

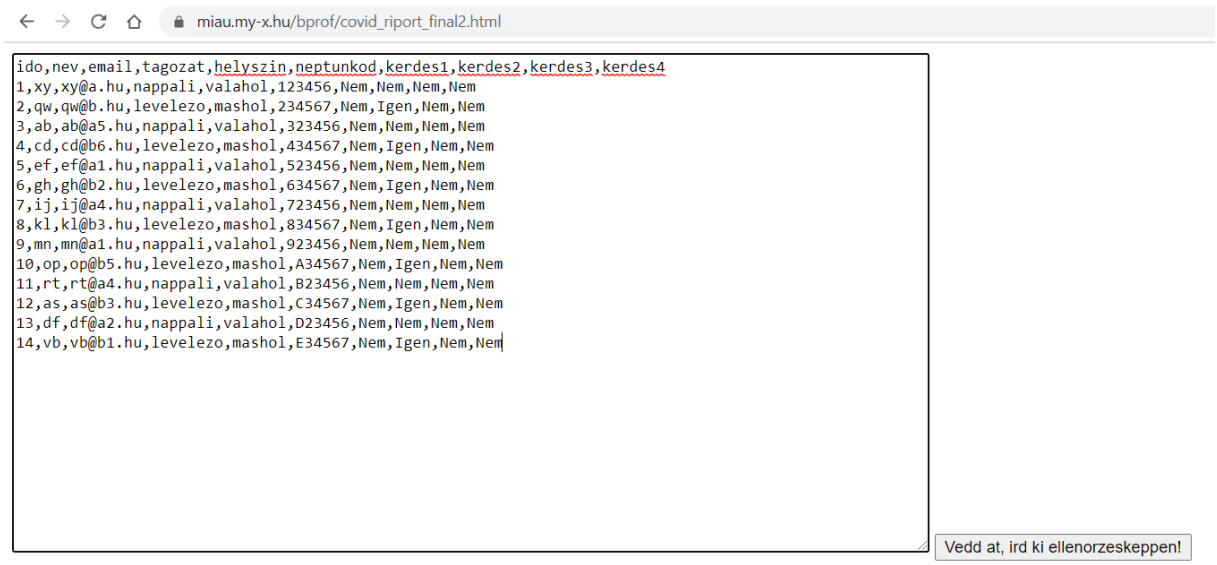
- Arra a szakértői kérdésre, vajon a laikus a keresés közben beazonosította-e a toString() és/vagy a join() megoldásokat a válasz határozottan nemleges volt mindkét opció esetére.
- Arra a kérdésre, hogy a laikus tisztában van-e azzal, hogy mi a különbség a res, a res2 és a res3 tartalma között, érdemi válasz nem érkezett.

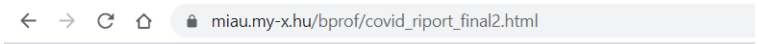
ido	nev	email	tagozat	helyszin	neptunkod	kerdes1	kerdes2	kerdes3	kerdes4
1	xy	xy@a.hu	nappali	valahol	123456	Nem	Nem	Nem	Nem
2	qw	qw@b.hu	levelezo	mashol	234567	Nem	Igen	Nem	Nem

ido,nev,email,tagozat,helyszin,neptunkod,kerdes1,kerdes2,kerdes3,kerdes4,1,xy,xy@a.hu,nappali,valahol,123456,Nem,Nem,Nem,Nem,2,qw,qw@b.hu,levelezo,mashol,234567,Nem,Igen,Nem,Nem,
ido,nev,email,tagozat,helyszin,neptunkod,kerdes1,kerdes2,kerdes3,kerdes4,1,xy,xy@a.hu,nappali,valahol,123456,Nem,Nem,Nem,Nem,2,qw,qw@b.hu,levelezo,mashol,234567,Nem,Igen,Nem,Nem,
2,qw,qw@b.hu,levelezo,mashol,234567,Nem,Igen,Nem,Nem

- Miért kellett a res3-at egyáltalán bevezetni? A válasz: valahol felmerült az Internetes anyagokban az, hogy nem csak egydimenziós, hanem két-dimenziós tömbök is léteznek.
  - Arra a szakértői felvetésre, miszerint biztos-e a laikus abban, hogy a res3 tartalma immár egy mátrix, a válasz egyértelmű igen volt, hiszen a HTML-oldalon ott látható a táblázat, s az a res3 maga.
  - Arra a szakértői kérdésre, hogy a res3 nélkül is lehetséges-e a HTML-táblázat előállítása, a válasz igenlő volt, hiszen a `res2[10*j+i];` a lényeg!
  - Arra a szakértői kérdésre, mit is jelent a `var res3 = [[],[]];` , a bizonytalan válasz az volt, hogy így lehet a két-dimenziós tömböt kialakítani…
- Lehetne-e kevesebb változóval ugyanezt a hatást elérni? Az elgondolkodó válasz nem volt más, mint az, hogy most, a szakértői kérdések után, vélhetően igen…

A laikus ezek után új kódot alkotott a fenti szakértői impulzusok figyelembe vételével:





15
10

ido	nev	email	tagozat	helyszin	neptunkod	kerdes1	kerdes2	kerdes3	kerdes4
1	xy	xy@a.hu	nappali	valahol	123456	Nem	Nem	Nem	Nem
2	qw	qw@b.hu	levelezo	mashol	234567	Nem	Igen	Nem	Nem
3	ab	ab@a5.hu	nappali	valahol	323456	Nem	Nem	Nem	Nem
4	cd	cd@b6.hu	levelezo	mashol	434567	Nem	Igen	Nem	Nem
5	ef	ef@a1.hu	nappali	valahol	523456	Nem	Nem	Nem	Nem
6	gh	gh@b2.hu	levelezo	mashol	634567	Nem	Igen	Nem	Nem
7	ij	ij@a4.hu	nappali	valahol	723456	Nem	Nem	Nem	Nem
8	kl	kl@b3.hu	levelezo	mashol	834567	Nem	Igen	Nem	Nem
9	mn	mn@a1.hu	nappali	valahol	923456	Nem	Nem	Nem	Nem
10	op	op@b5.hu	levelezo	mashol	A34567	Nem	Igen	Nem	Nem
11	rt	rt@a4.hu	nappali	valahol	B23456	Nem	Nem	Nem	Nem
12	as	as@b3.hu	levelezo	mashol	C34567	Nem	Igen	Nem	Nem
13	df	df@a2.hu	nappali	valahol	D23456	Nem	Nem	Nem	Nem
14	vb	vb@b1.hu	levelezo	mashol	E34567	Nem	Igen	Nem	Nem

```
<!DOCTYPE HTML>
<html>
<head>
<textarea id="myText" rows="25" cols= 100"></textarea>
<button onclick="myFunction2()">Vedd at, ird ki ellenorzeskeppen!</button>
<script>
function myFunction2() {
  var x = document.getElementById("myText").value;
  var res = x.split("\n");
  var sor = x.split("\n").length;
  res = res+",";
  var res2 = res.split(",");
  var oszlop = (res.split(",").length-1)/sor;
document.write( sor + "<br/>");
document.write( oszlop + "<br/>");
document.write( "<table border='1'>" );
document.write( "<tbody>" );
for ( j=0; j<sor; ++j )
{
document.write( "<tr>" );
for ( i=0; i<oszlop; ++i)  { document.write( "<td>"+res2[oszlop*j+i]+"</td>"
);  }
document.write( "</tr>" );
}
document.write( "</tbody>" );
document.write( "</table>" );

}
</script>

</body>
</html>
```

Ezen a ponton immár csak két feladat maradt, melyek mindegyike a HA()-feltételvizsgálatot igényli. Az egyik nem más, mint hogy a fejléc mindenkor legyen adott, vagyis az 1. sor, ill. a res2 első 10 eleme mindenkor legyen kiírva. A másik, hogy csak azok a sorok legyenek kiírva, ahol adott kérdés kapcsán (jelen esetben a kerdes2 esetében) van „Igen" válasz.

Forrás: https://www.w3schools.com/jsref/jsref_if.asp

```
document.write( "<table border='1'>" );
document.write( "<tbody>" );
for ( j=0; j<sor; ++j )
{
document.write( "<tr>" );
for ( i=0; i<oszlop; ++i)
{
if ( j < 1 ) {
document.write( "<td>"+res2[oszlop*j+i]+"</td>" );
}

if ( res2[oszlop*j+7] === "Igen" ) {
document.write( "<td>"+res2[oszlop*j+i]+"</td>" );
}

}
document.write( "</tr>" );
}
document.write( "</tbody>" );
document.write( "</table><br/>" );
```

A `res2[oszlop*j+7] === "Igen"` esetében a +7 a 0-tól induló sorszámozásnál a 8. oszlopra utal statikusan, ahol a 7-es konstans érték egy következő változatban a HTML-nyitóképből megfelelő input lehetőség mellett egy, a felhasználó által kijelölhető kérdés mögötti konstans is lehetne.

Végeredmények: https://miau.my-x.hu/bprof/covid_riport_final3_democsv.html + https://miau.my-x.hu/bprof/csv_demo, ill. https://miau.my-x.hu/bprof/covid_riport_final3_real.html (olyan CVS-k számára, ahol minden tartalomegység idézőjelek között van).

SPLIT-Alternatívák: Excel-cellatartomány esetén "\n" helyett " String.fromCharCode(10), ill. "," helyett "\t" (a horizontális tabulatort jelzendő).

15
10

ido	nev	email	tagozat	helyszin	neptunkod	kerdes1	kerdes2	kerdes3	kerdes4
2	qw	qw@b.hu	levelezo	mashol	234567	Nem	Igen	Nem	Nem
4	cd	cd@b6.hu	levelezo	mashol	434567	Nem	Igen	Nem	Nem
6	gh	gh@b2.hu	levelezo	mashol	634567	Nem	Igen	Nem	Nem
8	kl	kl@b3.hu	levelezo	mashol	834567	Nem	Igen	Nem	Nem
10	op	op@b5.hu	levelezo	mashol	A34567	Nem	Igen	Nem	Nem
12	as	as@b3.hu	levelezo	mashol	C34567	Nem	Igen	Nem	Nem
14	vb	vb@b1.hu	levelezo	mashol	E34567	Nem	Igen	Nem	Nem

ido	nev	email	tagozat	helyszin	neptunkod	kerdes1	kerdes2	kerdes3	kerdes4
1	xy	xy@a.hu	nappali	valahol	123456	Nem	Nem	Nem	Nem
2	qw	qw@b.hu	levelezo	mashol	234567	Nem	Igen	Nem	Nem
3	ab	ab@a5.hu	nappali	valahol	323456	Nem	Nem	Nem	Nem
4	cd	cd@b6.hu	levelezo	mashol	434567	Nem	Igen	Nem	Nem
5	ef	ef@a1.hu	nappali	valahol	523456	Nem	Nem	Nem	Nem
6	gh	gh@b2.hu	levelezo	mashol	634567	Nem	Igen	Nem	Nem
7	ij	ij@a4.hu	nappali	valahol	723456	Nem	Nem	Nem	Nem
8	kl	kl@b3.hu	levelezo	mashol	834567	Nem	Igen	Nem	Nem
9	mn	mn@a1.hu	nappali	valahol	923456	Nem	Nem	Nem	Nem
10	op	op@b5.hu	levelezo	mashol	A34567	Nem	Igen	Nem	Nem
11	rt	rt@a4.hu	nappali	valahol	B23456	Nem	Nem	Nem	Nem
12	as	as@b3.hu	levelezo	mashol	C34567	Nem	Igen	Nem	Nem
13	df	df@a2.hu	nappali	valahol	D23456	Nem	Nem	Nem	Nem
14	vb	vb@b1.hu	levelezo	mashol	E34567	Nem	Igen	Nem	Nem

Az eredmény-HTML esetén természetesen csak a felső táblázat a releváns, de az alsó szükséges a gyors vizuális ellenőrzés érdekében itt és most. A felső két szám értelemszerűen a sorok száma (fejléccel együtt), s az oszlopok száma.

További szakértői vélemények (utólag):

1.

var res = x.split("\n");

var sor = x.split("\n").length;

Ilyen ismétléseket nem ajánlatos írni, főleg ha a közös részkifejezés bármi okból amúgy is saját változóba kerül. Inkább: pl.

var res = x.split("\n");

var sor = res.length;

Ezzel nem kell pl. optimalizálásra támaszkodni az ismételt kiértékelés elkerülése kapcsán.

2.

if (j < 1) {

document.write( );

}

if (res2[oszlop*j+7] === "Igen") {

document.write( );

}

Nem független feltételeket nem célszerű külön if-ek formájában megírni, mert mindkét ág kiértékelésre kerülhet elméletileg, még ha most a mintaadatokon ez nem is fordult elő. Kifejezőbb lenne "else if" használata (vagy éppen continue, de az más hibák forrásává válhat).

A mini-projekt folytatásának lehetőségei

Az eddig leírt elvárások és megoldások kapcsán a következő (nem teljeskörű) fejlesztési lehetőség megoldásával is lehet számolni:

- Felkészülés olyan tartalmakra, melyek idézőjelek között ugyan, de vesszőt (delimitert) tartalmaznak…
- Felkészülés Excel-ből átvett adatok kezelésére (ahol más a sorvégjel, s más lehet a delimiter, hiszen át lehet venni Excel-ből egy oszlopnyi, még több oszloppá át nem alakított adatot, s át lehet venni Excel-ből már több oszloppá alakított adatvagyont is)…
- Felkészülés arra az esetre, ha az utolsó adat után még van egy sorvégjel…
- Felkészülés az összes (pl. eddig felsorolt) elképzelhető variáns automatikus felismerésére, egy rendszerben való kezelésére…
- Felkészülés a „talán” opció kezelésére…
- Felkészülés a karakterkódolások kezelésére…
- Felkészülés a riportálandó kérdés felhasználó általi előzetes kiválasztására…
- Felkészülés egy riport után új kérdés kiválasztására és új riport generálására – új adatátadás nélkül…
- …

Konklúziók

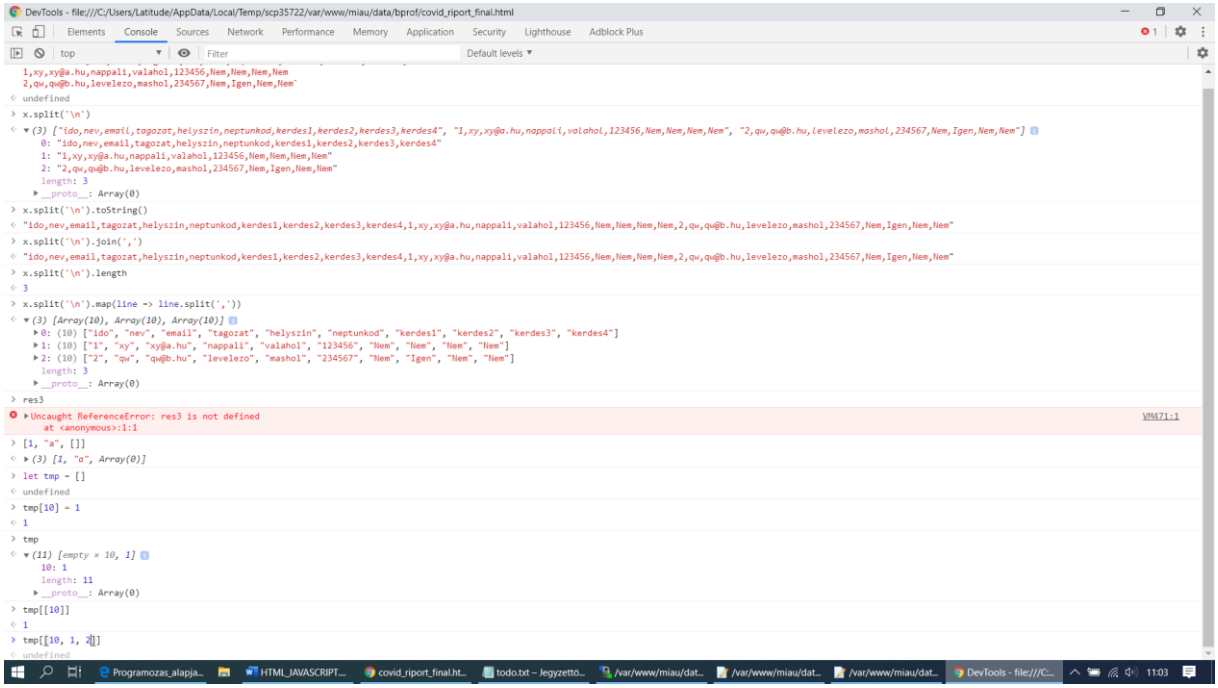
A laikus kódkölcsönzések és kódértelmezések érdemi debug-támogatás és a változók pontos tartalmáról alkotott ellenőrzött elképzelések nélkül is meglepően hatásosan vezettek eredményre – természetesen számos kockázatot rejtve magukban.

A laikus már a szakértői kérdések alapján is képes volt új gondolatokat, megoldási stratégiákat kialakítani, korábbi kényszerpályákat elhagyni.

A kódkölcsönzés stratégiája hatékonynak bizonyult, mert a laikus tényleg alig tudott meg többet a programozásról, mint ami a konkrét feladat megoldásához neki feltétlenül kellett – s természetesen ez egyszerre előny és egyszerre kockázatok forrása, mely kockázatok adott feladat kapcsán lehet, hogy sosem lépnek fel, de az egyik megoldás után a másik megoldásra való áttéréskor a félreértések komoly zavarok forrásával válhatnak (vö. tömbök fizikai, vizualizált tartalmának esetleges eltérései, ill. tömbök és string-ek értelmezésének zavarai).

Az esettanulmány 2. részében majd bemutatásra kerül, miként lehet a debug érdekében a JAVASCRIPT-kódot pl. önálló .js-állományként tárolva és a böngésző F12-es szolgáltatásait igénybe véve a hatásosságot nagyobb biztonság mellett garantálni…

Melléklet – szakértői aktivitások nyomai




```
<!DOCTYPE HTML>
<html>
<head>
<textarea id="myText" rows="5" cols= 100"></textarea>
<button onclick="myFunction2()">Vedd at, ird ki ellenorzeskeppen!</button>
<script>
function myFunction2() {
  var x = document.getElementById("myText").value;
  var res = x.split("\n");
  res = res+",";
  var res2 = res.split(",");
  var res3 = [[],[],[]];
document.write( "<table border='1'>" );
document.write( "<tbody>" );
for ( j=0; j<3; ++j )
{
document.write( "<tr>" );
for  (  i=0;  i<10;  ++i)      {  res3[j][i]=res2[10*j+i];  document.write(
"<td>"+res3[j][i]+"</td>" );  }
document.write( "</tr>" );
}
document.write( "</tbody>" );
document.write( "</table>" );

document.write( res +"<br/>");
document.write( res2 +"<br/>");
document.write( res3 +"<br/>");
}
</script>
</body>
</html>
```

ido	nev	email	tagozat	helyszin	neptunkod	kerdes1	kerdes2	kerdes3	kerdes4
1	xy	xy@a.hu	nappali	valahol	123456	Nem	Nem	Nem	Nem
2	qw	qw@b.hu	levelezo	mashol	234567	Nem	Igen	Nem	Nem

ido,nev,email,tagozat,helyszin,neptunkod,kerdes1,kerdes2,kerdes3,kerdes4,1,xy,xy@a.hu,nappali,valahol,123456,Nem,Nem,Nem,Nem,2,qw,qw@b.hu,levelezo,mashol,234567,Nem,Igen,Nem,Nem,
ido,nev,email,tagozat,helyszin,neptunkod,kerdes1,kerdes2,kerdes3,kerdes4,1,xy,xy@a.hu,nappali,valahol,123456,Nem,Nem,Nem,Nem,2,qw,qw@b.hu,levelezo,mashol,234567,Nem,Igen,Nem,Nem,
ido,nev,email,tagozat,helyszin,neptunkod,kerdes1,kerdes2,kerdes3,kerdes4,1,xy,xy@a.hu,nappali,valahol,123456,Nem,Nem,Nem,Nem,2,qw,qw@b.hu,levelezo,mashol,234567,Nem,Igen,Nem,Nem