Kliensoldali webprogramozás csoportzh -- Progresszív fejlesztés

2022.03.01.

Horváth Győző

KERETPROGRAM

Tudnivalók

A feladat beküldésével az alább leírtakat megértettnek és elfogadottnak tekintjük annak a nevében, aki a megoldást beküldte.

```
<Hallgató neve>
<Neptun kódja>
Ezt a megoldást a fent írt hallgató küldte be és készítette
a Kliensoldali webprogramozás kurzus csoport ZH-jához.
Kijelentem, hogy ez a megoldás a saját munkám. Nem másoltam vagy
használtam harmadik féltől származó megoldásokat. Nem továbbítottam
megoldást hallgatótársaimnak, és nem is tettem közzé. Az Eötvös Loránd
Tudományegyetem Hallgatói Követelményrendszere (ELTE szervezeti és
működési szabályzata, II. Kötet, 74/C. §) kimondja, hogy mindaddig,
amíg egy hallgató egy másik hallgató munkáját - vagy legalábbis annak
jelentős részét - saját munkájaként mutatja be, az fegyelmi vétségnek számít.
A fegyelmi vétség legsúlyosabb következménye a hallgató elbocsátása az
egyetemről.
```

1. Képek késleltetett betöltése (image-lazy-loading, 6 pont)

Egy hosszú oldalon szeretnénk a képeket csak akkor megjeleníteni, amikor azok a képernyő területére beérnek, így minimalizálva az oldal betöltésekor elküldött HTTP kérések számát. A képeknél a kép forrása a `data-src` attribútumban van tárolva, innen kell megjelenítéskor az értékét az `src` attribútumba másolni.

- a. (2 pont) Ha egy kép a megjelenített területre (viewportra) kerül, akkor a konzolra írjuk a kép `data-src` attribútumát!
- b. (1 pont) Ebben az esetben másoljuk a `data-src` attribútum értékét az `src` tulajdonságába.
- c. (1 pont) Ugyanekkor adjuk hozzá a `loaded` stílusosztályt is!
- d. (2 pont) Legyen a megoldásod hatékony, azaz ha lehet, minél kevesebbszer meghívódó eseménnyel operáljon!

 Képernyőkép a működésről

2. Automatikusan méreteződő beviteli mező (2-elastic-textarea, 7 pont)

Okosíts fel progresszíven egy textarea elemet úgy, hogy mindig olyan magas legyen, hogy a benne levő szöveg gördítősáv nélkül beleférjen! Zárd egységbe a megoldást, hogy osztályként is használható legyen majd később!

- a. (3 pont) Az első textareában való gépeléskor (`input` esemény) állítsd a textarea sorainak számát 1-re (`rows` tulajdonság), majd míg az `isScrolling` függvény igazat ad, növeld egyesével!
- c. (2 pont) Lehessen megadni a textarea maximális sor számát a `data-max-rows` tulajdonságban az osztály konstruktorában!
- d. Zárd egységbe a megoldást, hogy így is lehessen használni! (1 pont)

```
new ElasticTextarea(document.querySelector('[data-elastic]'))
```

- e. (2 pont) Oldd meg, hogy minden olyan textarea-ra működjön a megoldás, aminek van `data-elastic` attribútuma!

3. Szűrhető lista (7 pont)

Fejleszd fel az oldalon található listát úgy, hogy szűrhető legyen. Ehhez egy szöveges beviteli mezőt kell dinamikusan beszúrni a lista elemei elé. Progresszíven add hozzá ezt a funkciót a `HTMLUListElement` (`extends: 'ul'`) elemhez egy egyedire szabott webkomponensen keresztül. A felfejlesztett HTML szerkezet így néz ki:

```
<ul is='filterable-list'>
    <input type="text">
    <li></li>
    <!-- ... -->
</ul>
```

A mezőben gépelve pedig csak a megfelelő listaelemek jelennek meg.

- a. (2 pont) Szúrj be egy szöveges beviteli mezőt az oldalra!
- b. (1 pont) A szöveges beviteli mező a listaelemek elé kerüljön az `ul`-en belülre!
- d. (4 pont) A szöveges beviteli mezőben gépelve a listában csak azokat a listaelemeket jelenítsd meg, amelyek belső szövege (`innerText`) tartalmazza a szűrőmező értékét! A megjelenítéshez, eltüntetéshez használhatod a `hidden` tulajdonságát a DOM elemnek.