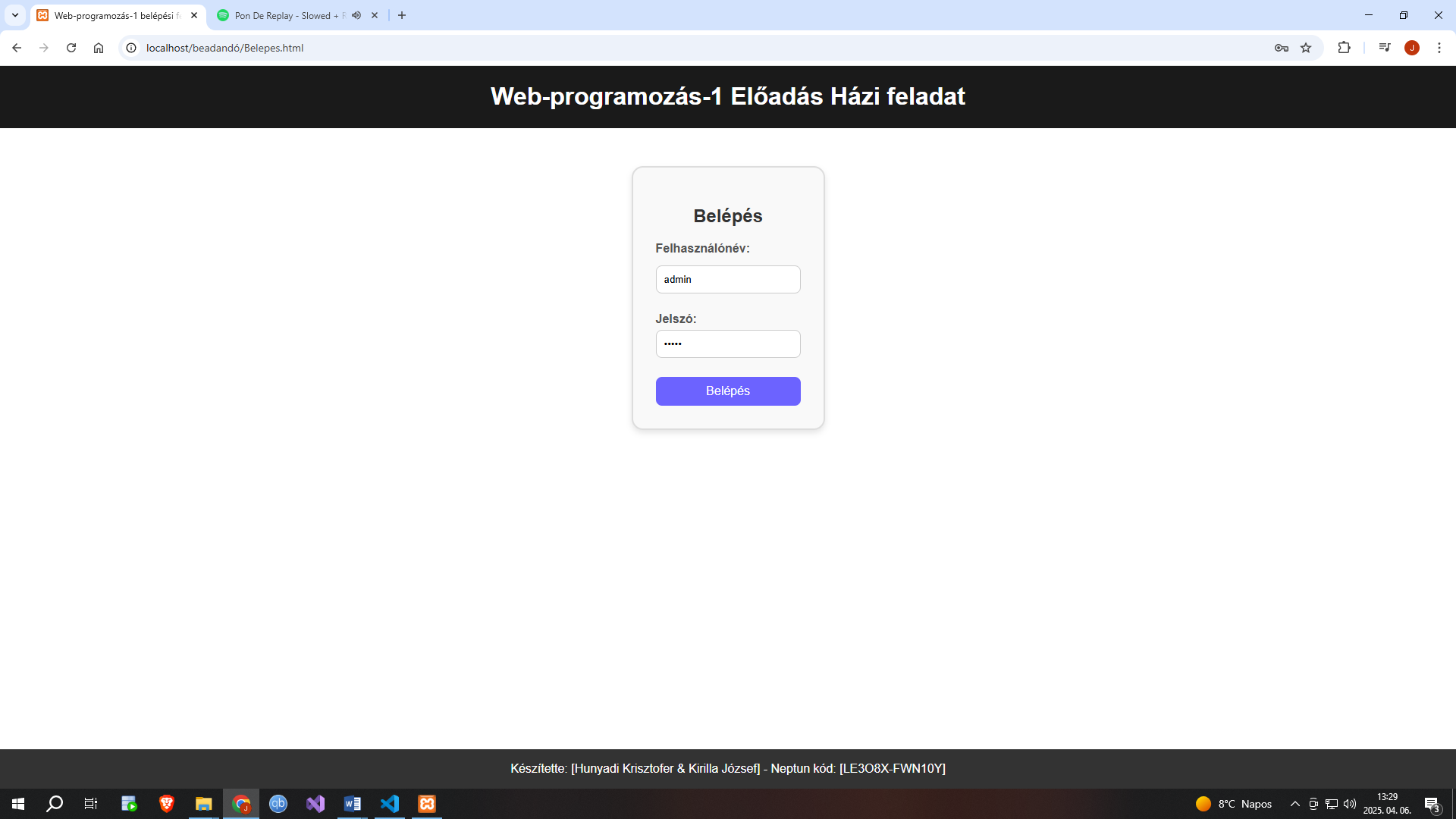
**Web-programozás-1 Előadás Házi feladat**

**Dokumentáció**

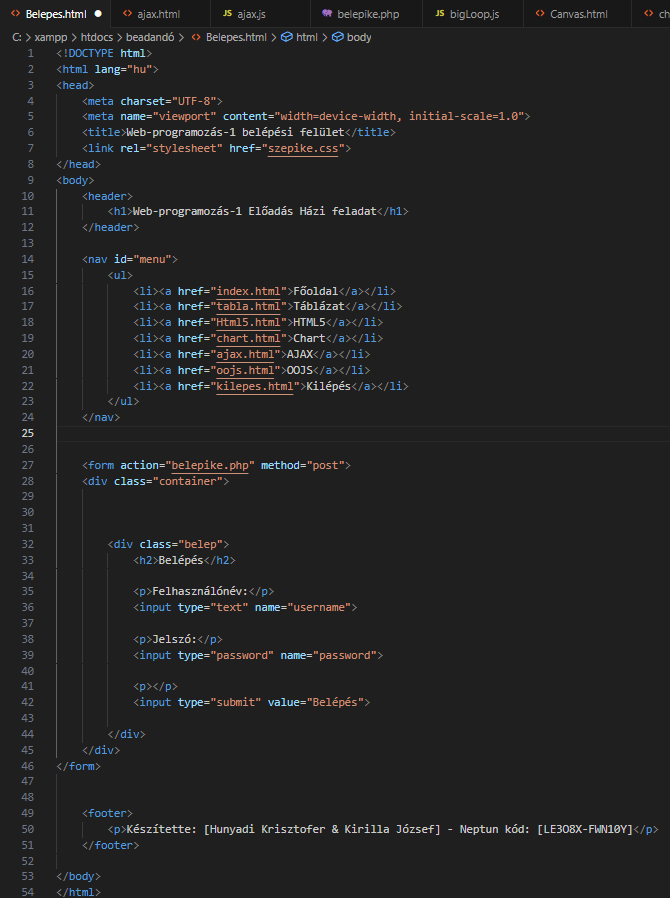
**Készítette: Kirilla József-(FWN10Y) és Hunyadi Krisztofer-(neptunkódod)**



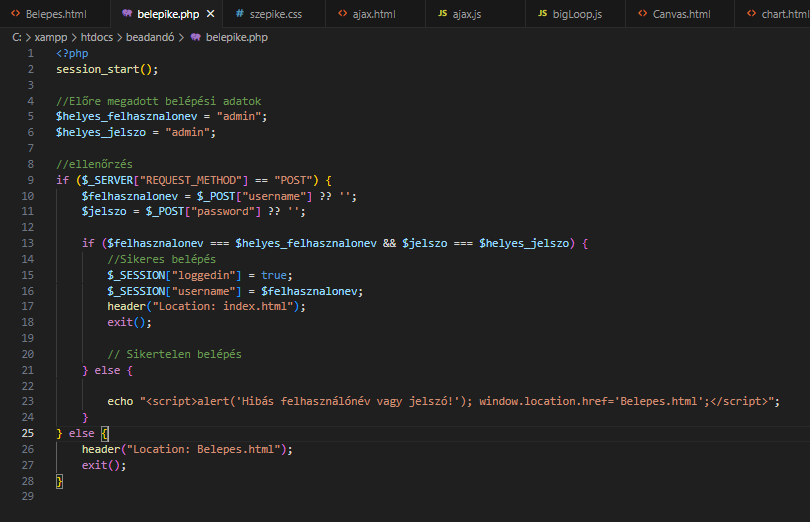
Első lapunk egy **plusz feladat** volt egy szimpla beléptető rendszerrel. **Felhasználónév**:”admin”

**jelszó**:”admin”

**belepes.html Kód:**

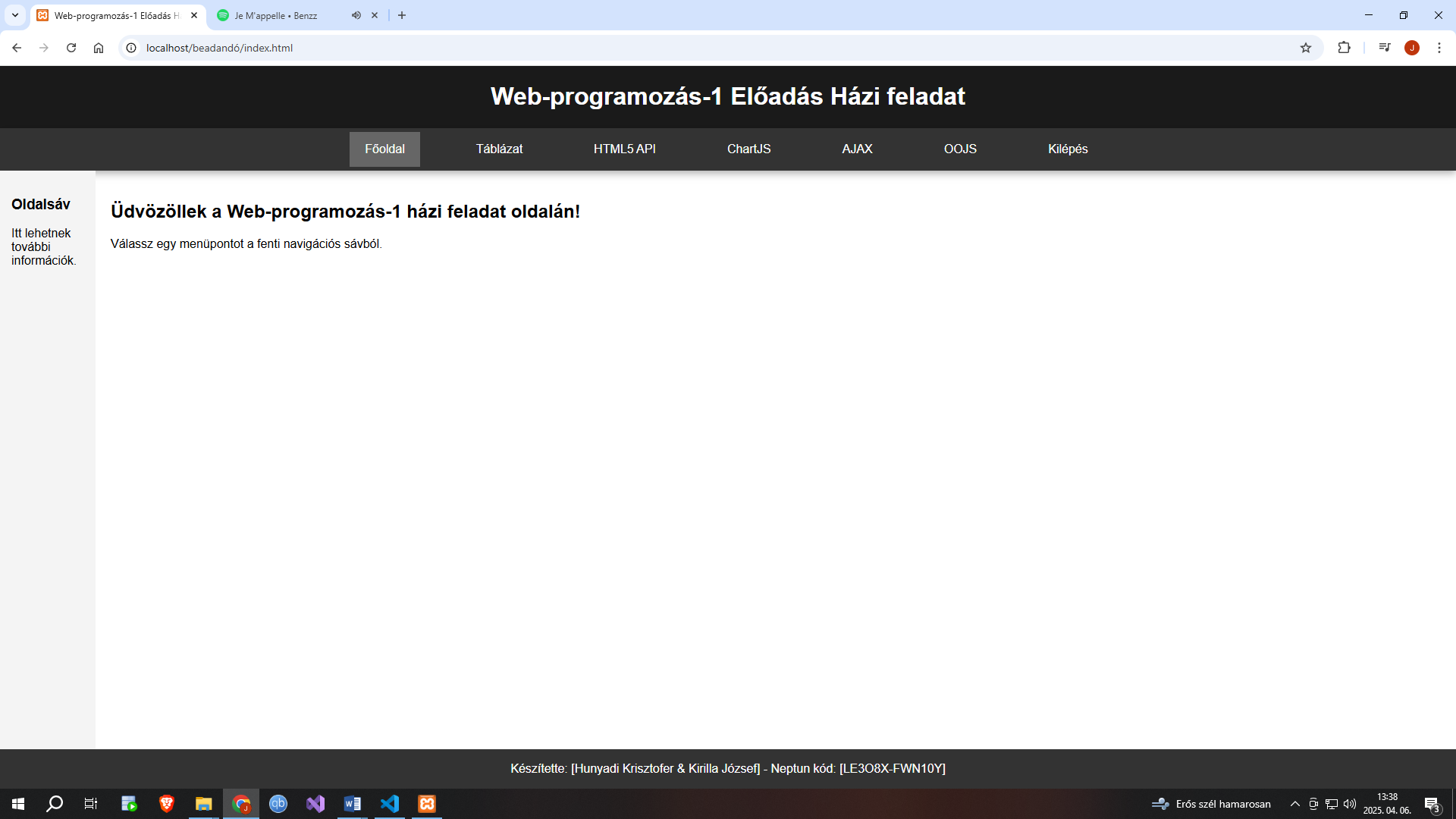


a jelszót a **belepike.php** ellenőrzi:



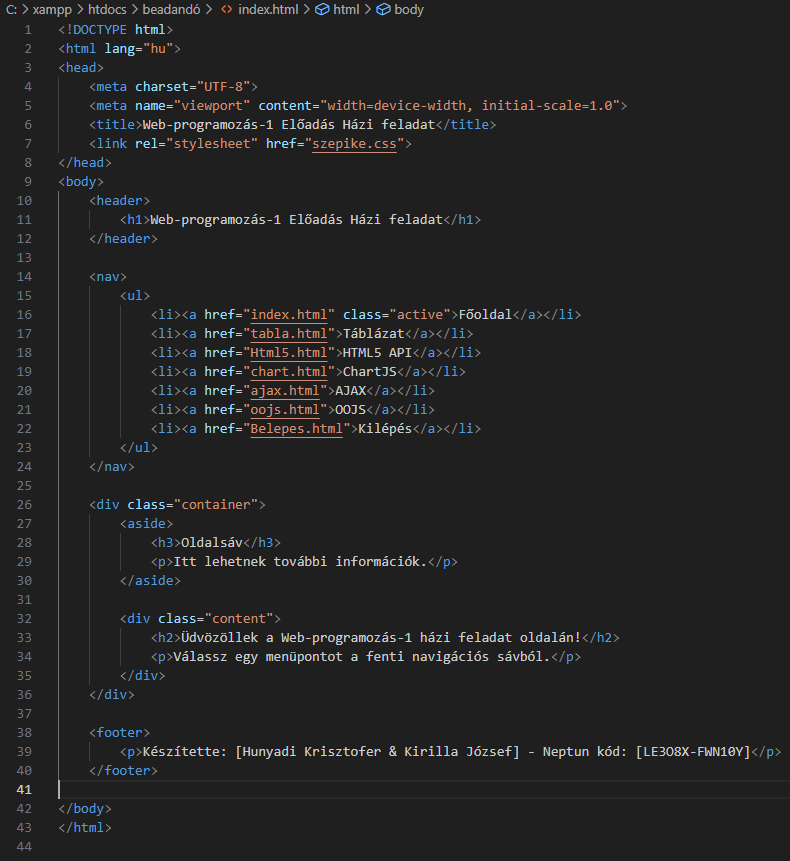
Értelemszerűen ha hibás a jelszó hibát ír. Ha pedig jól ütöttük be akkor továbbenged az **index.html**-re . (Továbbiakat ki kommenteltem)

**index.html**



(Az oldalsávot nem használtuk, igaz különösebb funkciója nincs. De ha későbbiekben szeretnénk bővíteni egy jó ötletnek tűnik)

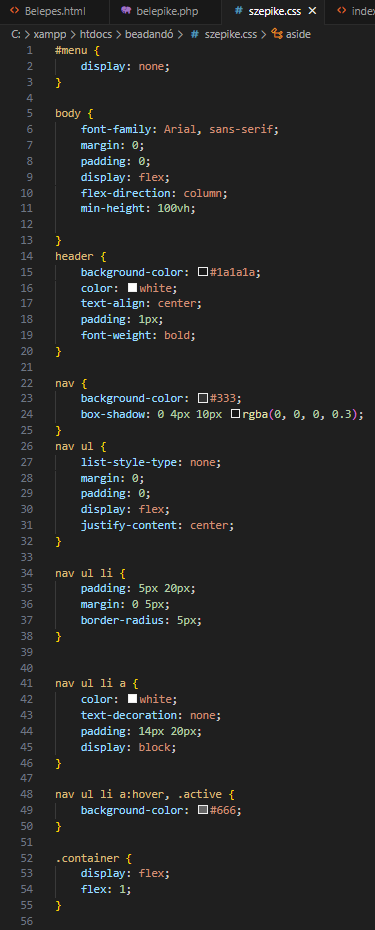
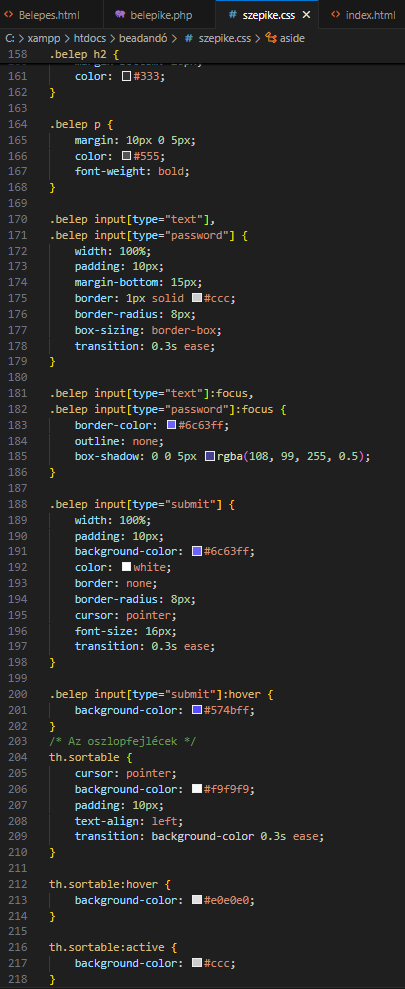
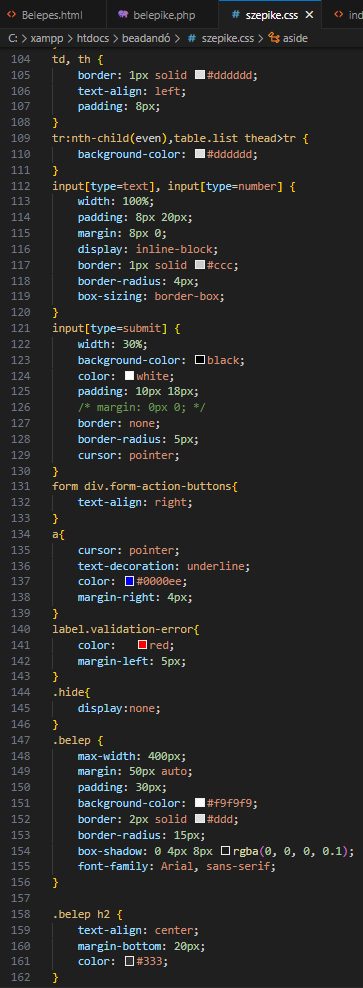
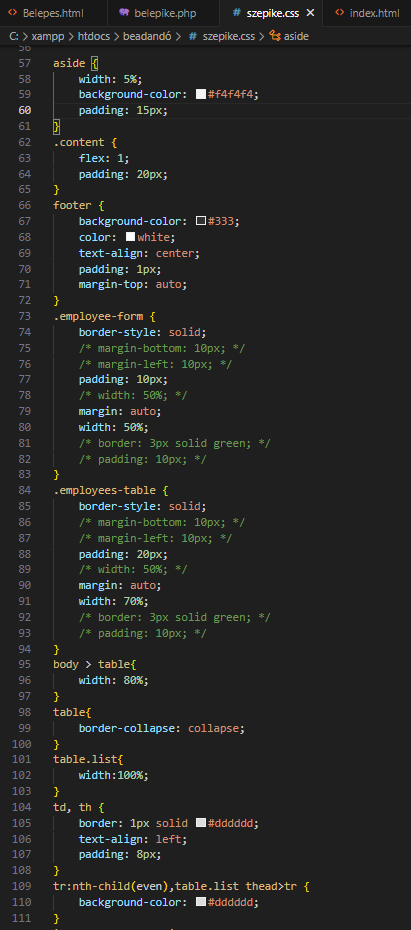
**index.html kód:**



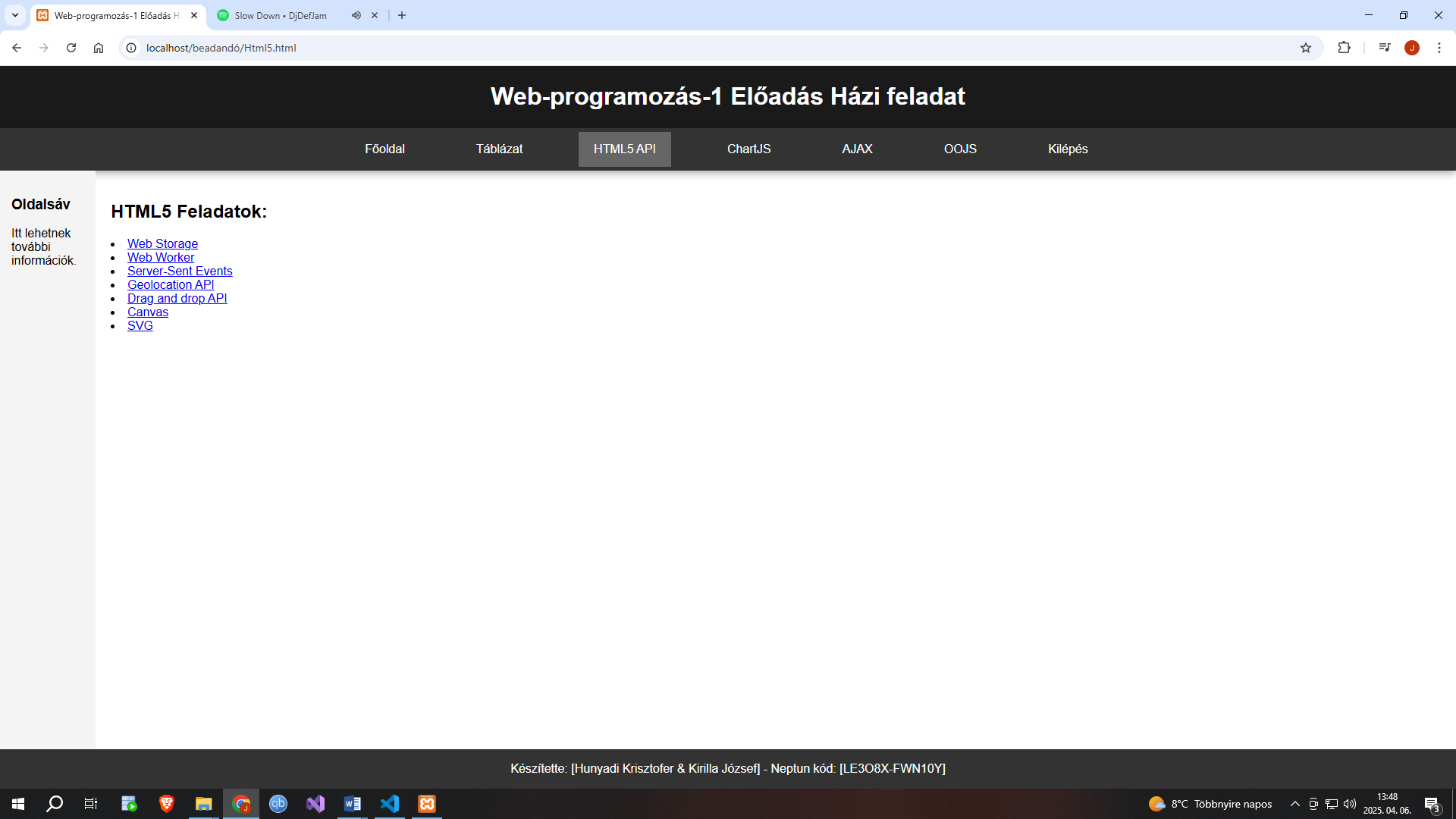
Teljes egészében a weboldal egy témára/stílusra épült rá amit a legkönnyebb az index.html-en lehet megmutatni. Ez magába foglalja a címeket a többi html meghívását, oldalsávot,üdvözlést, és a footert ahol a nevünk és a neptun kódunk található. Ezt a többi kódnál nem részletezném.

minden oldalunk a **szepike.css**-kódot használta design-nak

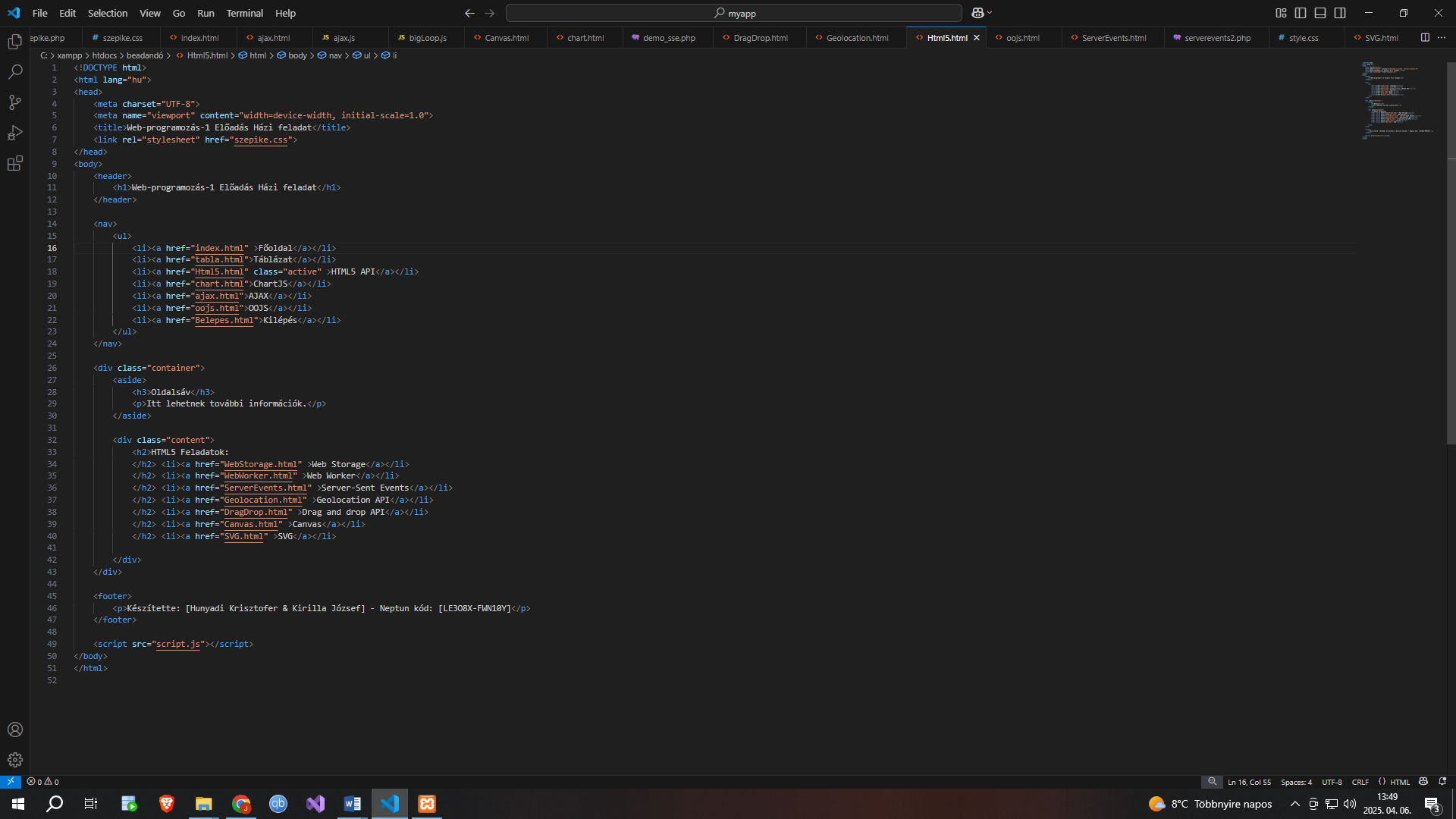
**szepike.css kódja:**

****

**HTML5-API**



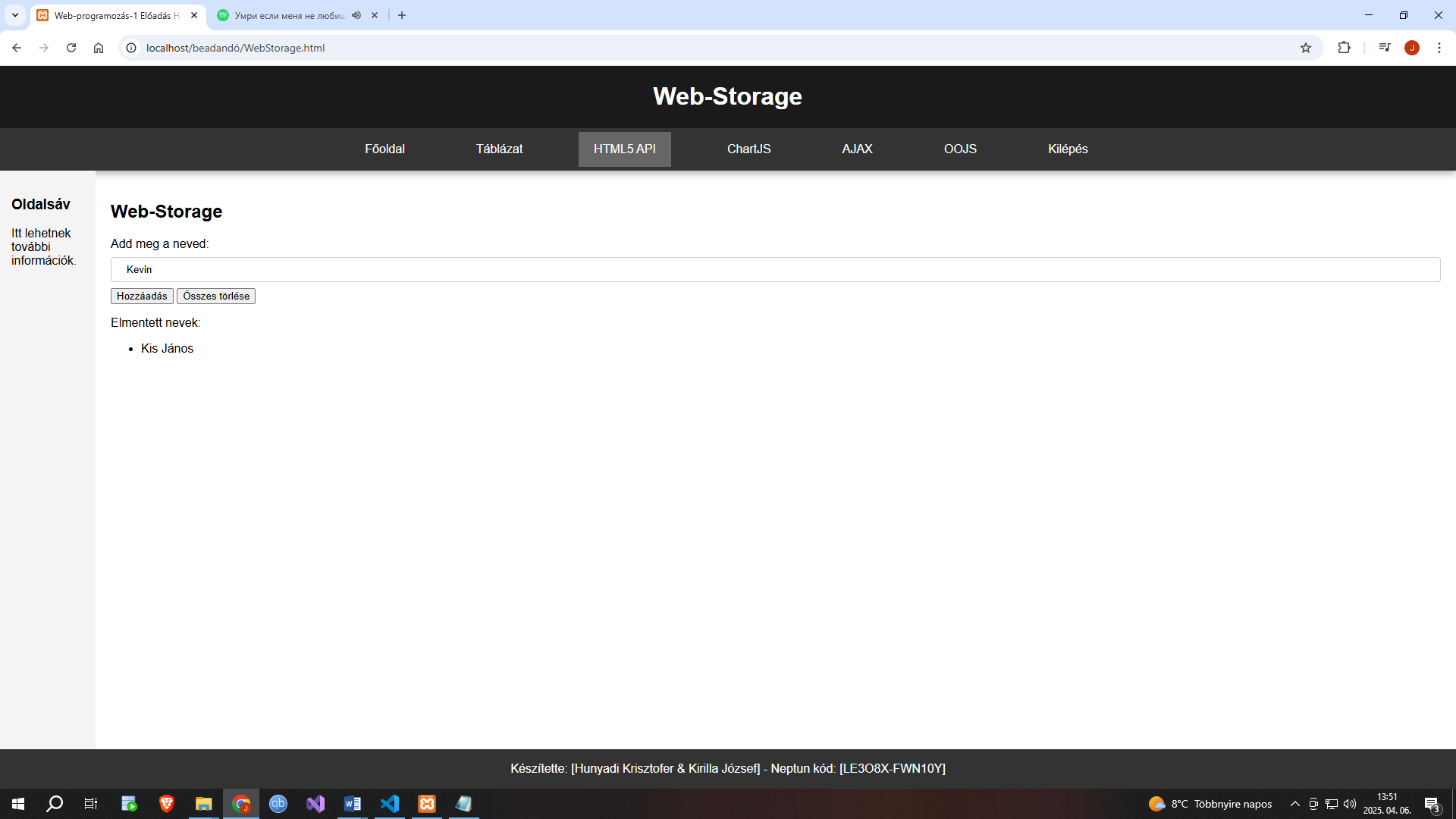
**HTML5-API kódja:**



**Ez tartalmaz egy listát különböző menüpontokkal. Mindegyik menüpont a saját egyedi menüjébe visz át**

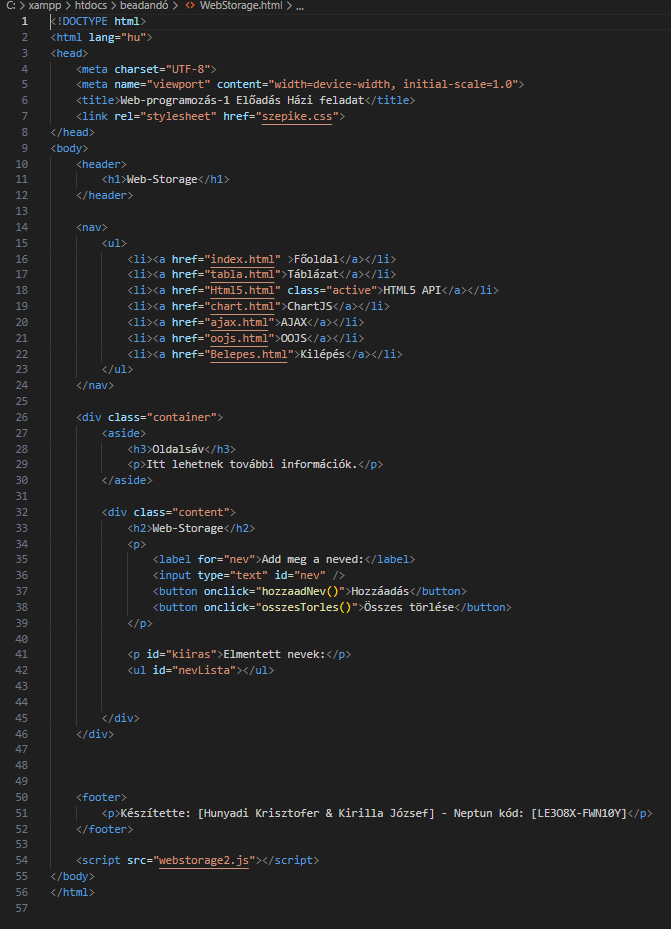
**HTML5-Külön menüpontjai:**

**Web Storage**



A Web-Storage Neveket tárol el Egyszerre akár többet is képes elmenteni. Majd egy gombbal tudja törölni az összeset

**WebStorage.html**



egyszerű gombos továbbküldés webstorage2.js-nek

**webstorage2.js**

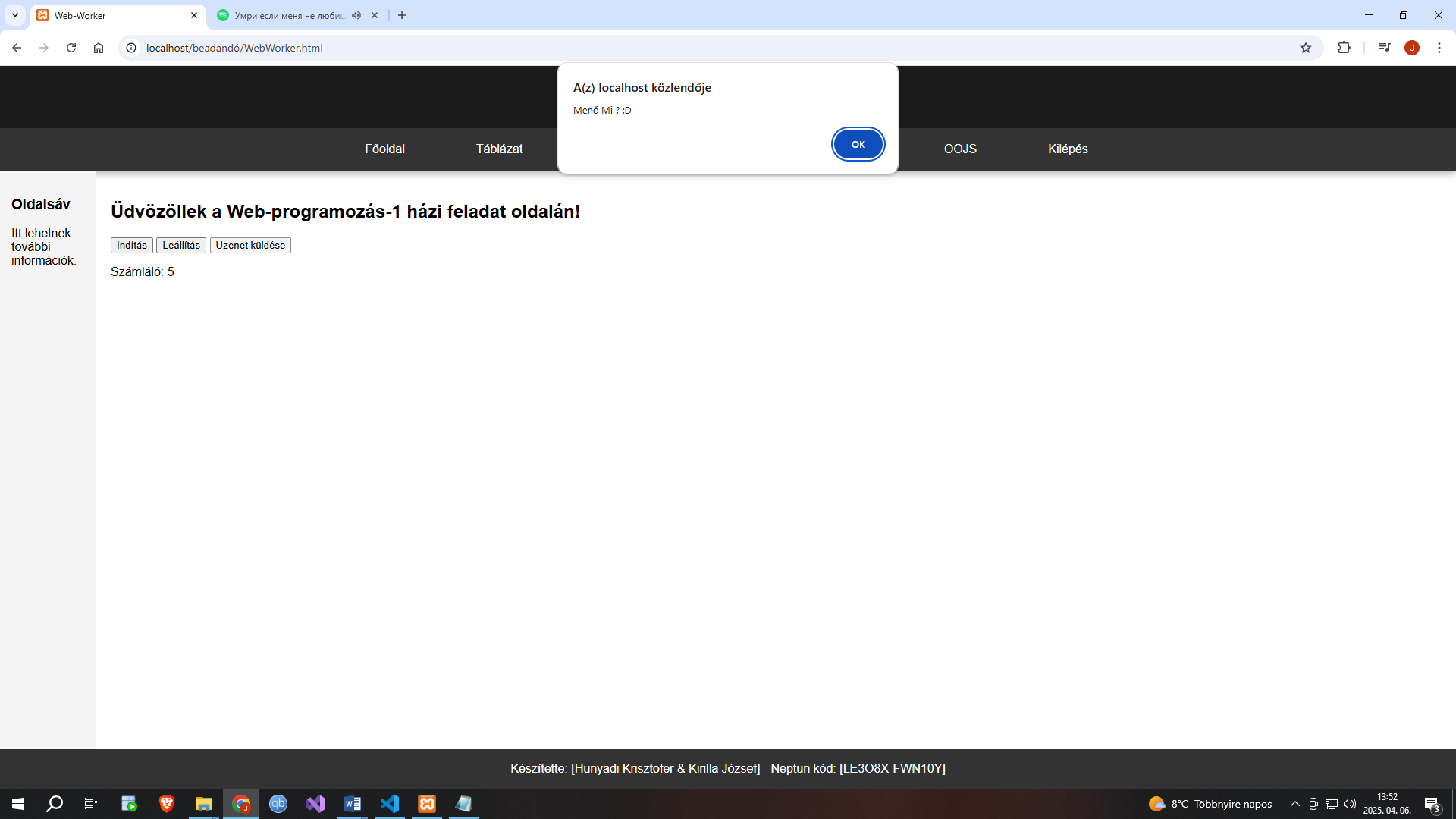


Ez a JavaScript kód neveket tárol el a böngészőben, és megjeleníti őket egy listában.Amikor betölt az oldal, betölti az eddig elmentett neveket.

Ha beírunk egy nevet és rákattintunk a hozzáadásra, elmenti a nevet a localStorage-be, és megjelenik a listában.

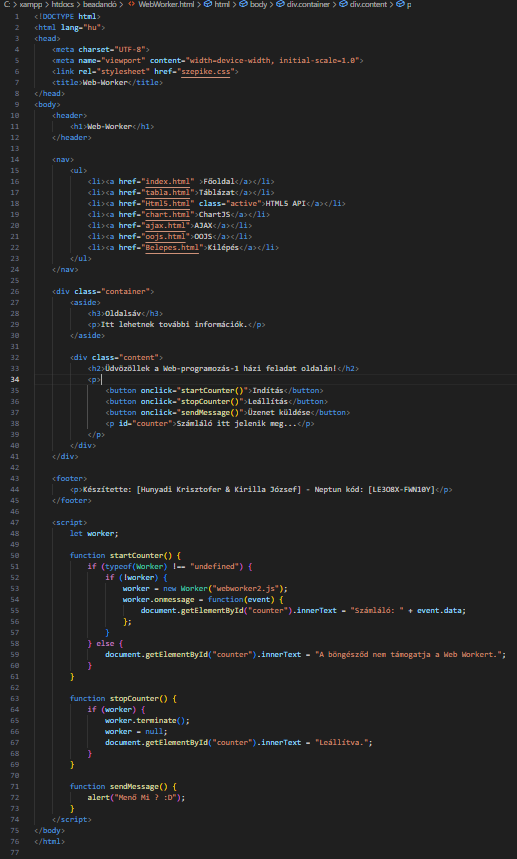
A nevek megmaradnak akkor is, ha az oldal frissítve lesz.

**Web Worker**

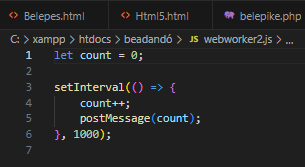


Ez a kód egy számlálót indít el egy Web Workerrel, ami külön szálon fut. Három gomb van: indítás, leállítás és üzenetküldés.A „Számláló” gombra kattintva elindul a számláló, és a worker visszaküldi az értékeket. A „Leállítás” leállítja a workert, az „Üzenet küldése” pedig csak küld neki egy szöveget

**Web Worker kódja:**

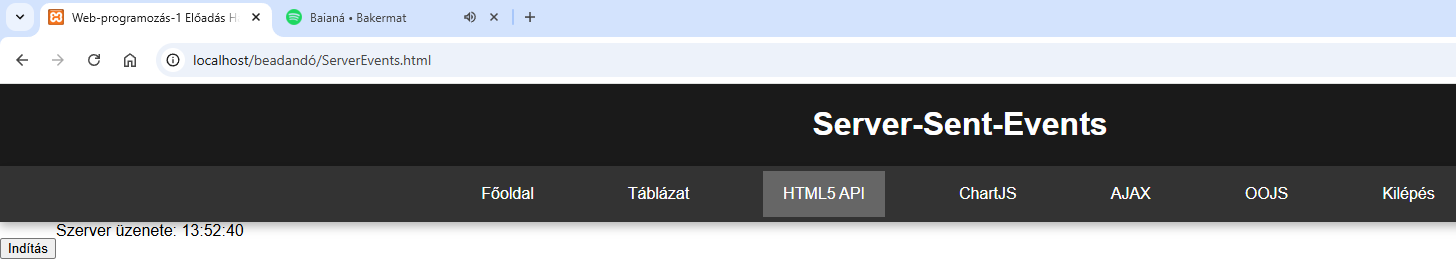
****

**web worker2.js**

****

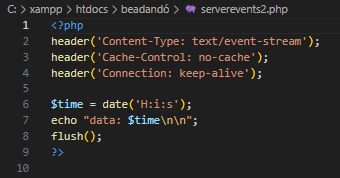
count változó 0-ról indul. Minden másodpercben (1000ms) növeli egyel a .postMessage(count) segítségével visszaküldi az aktuális értéket a főoldalnak. Ez adja a számláló működését, amit látunk.

**Server-Sent Events**

****

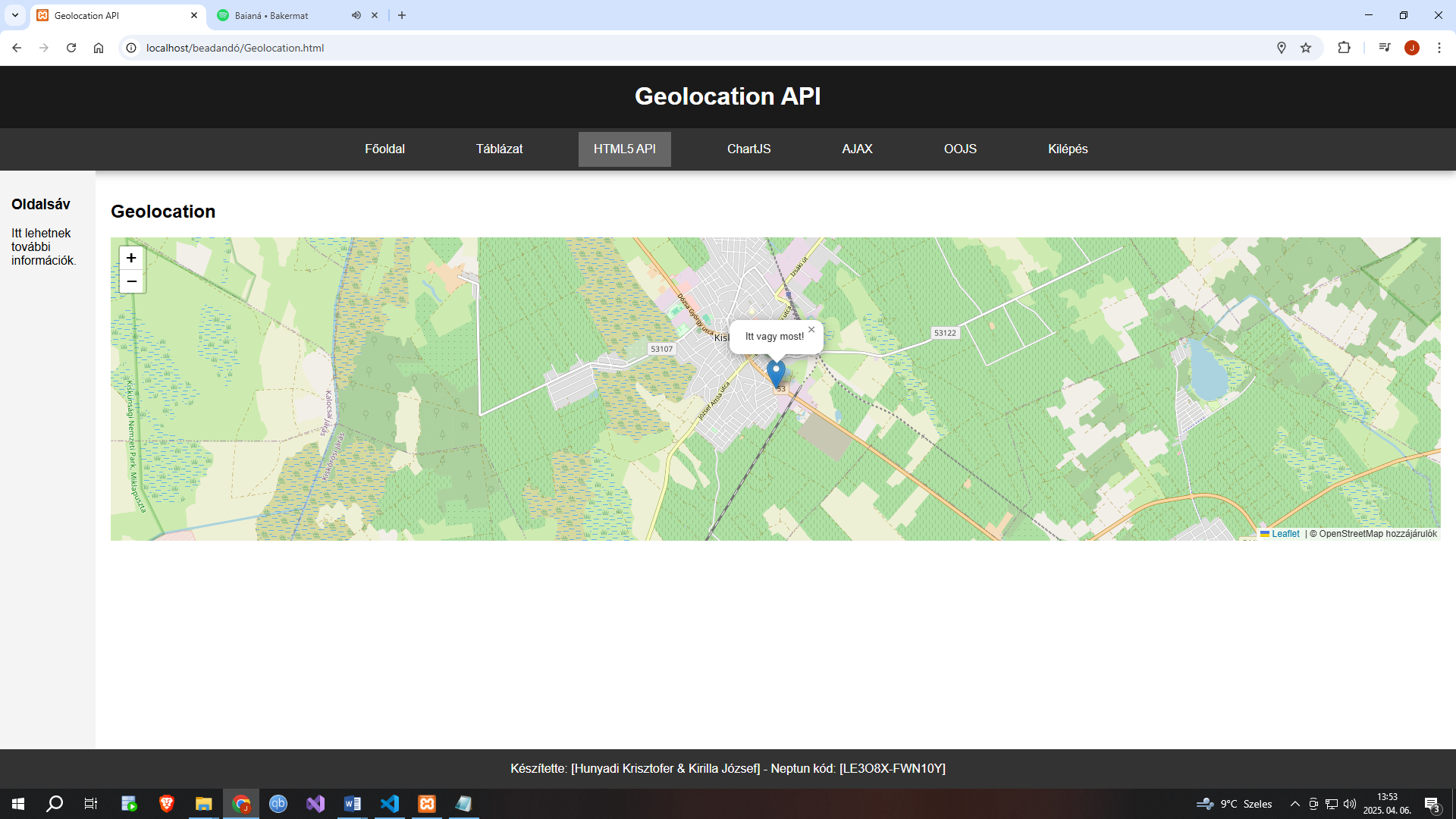
Van egy „Indítás” gomb, amire kattintva elkezdünk figyelni a szerverre (serverevents2.php).Ha jön valamilyen adat a szervertől, azt megjeleníti a <div id="sseData"> elemben.Ha a böngésző nem támogatja az SSE-t, kiírja, hogy nem működik.A kliens oldal folyamatosan automatikusan fogadja a szervertől jövő adatokat, frissítés nélkül.

**serverevents2.php**

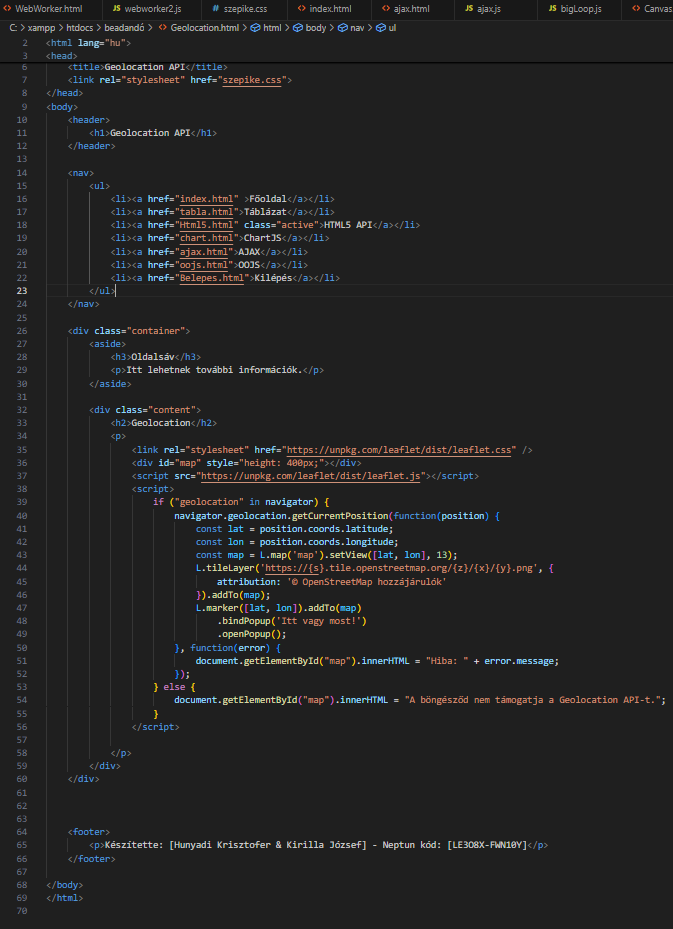


Beállítja a fejlécet, hogy SSE-ként küldje az adatot (text/event-stream).Lekéri az aktuális időt ($time).Kiírja az adatot data: formában, amit a JavaScript oldalon fogadunk.A flush() elküldi az adatot a böngészőnek.

**Geolocation API**

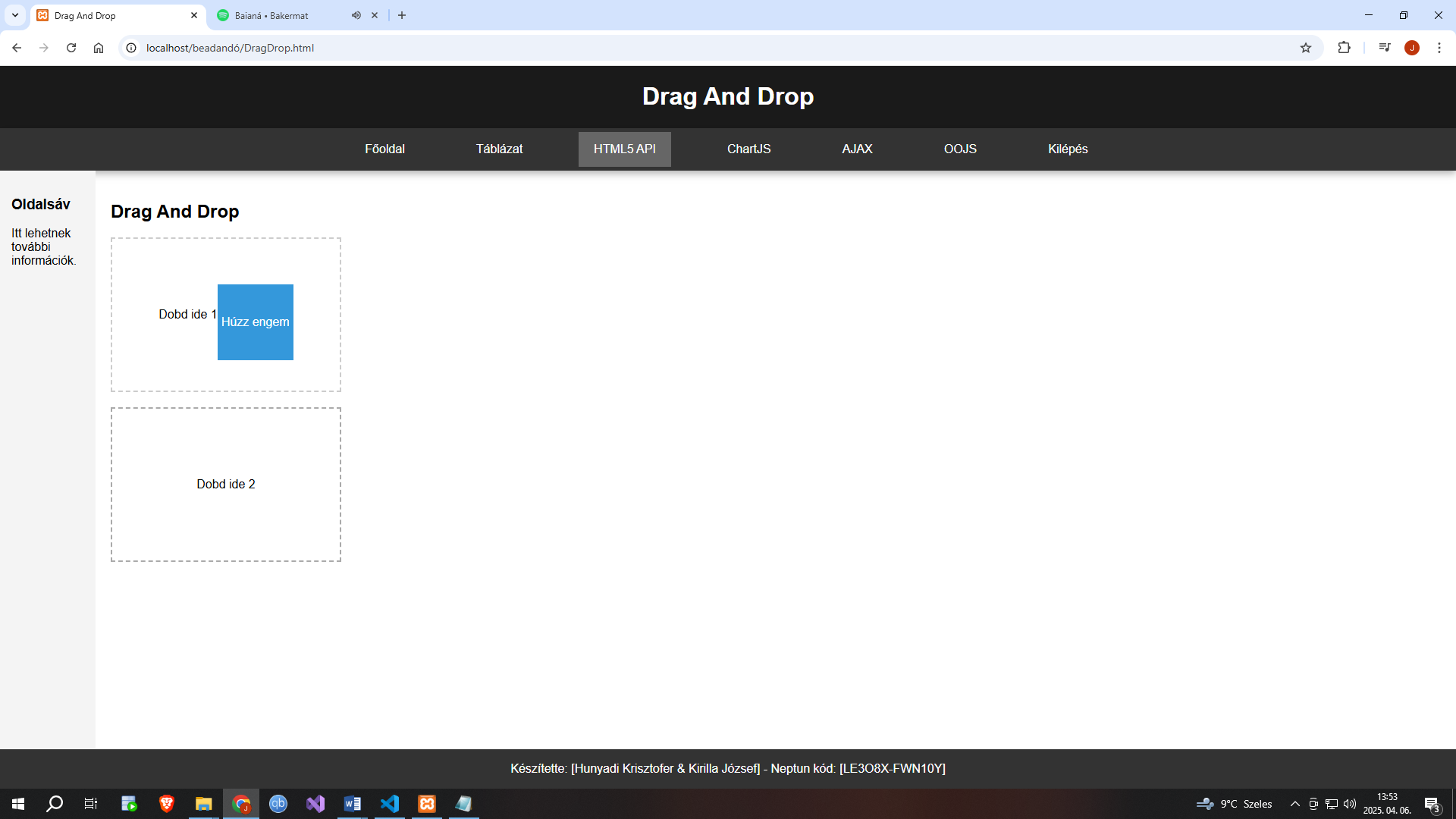
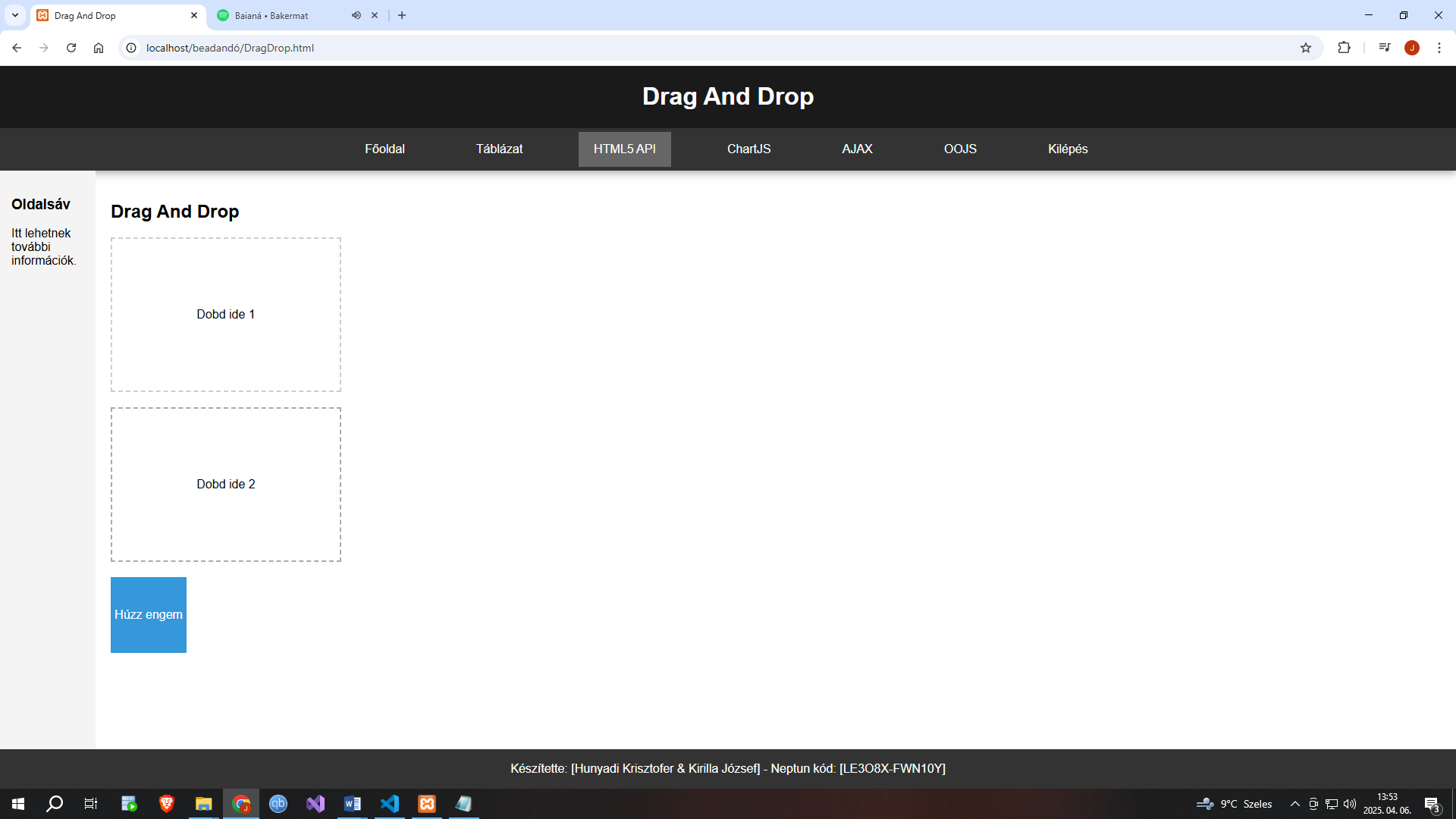


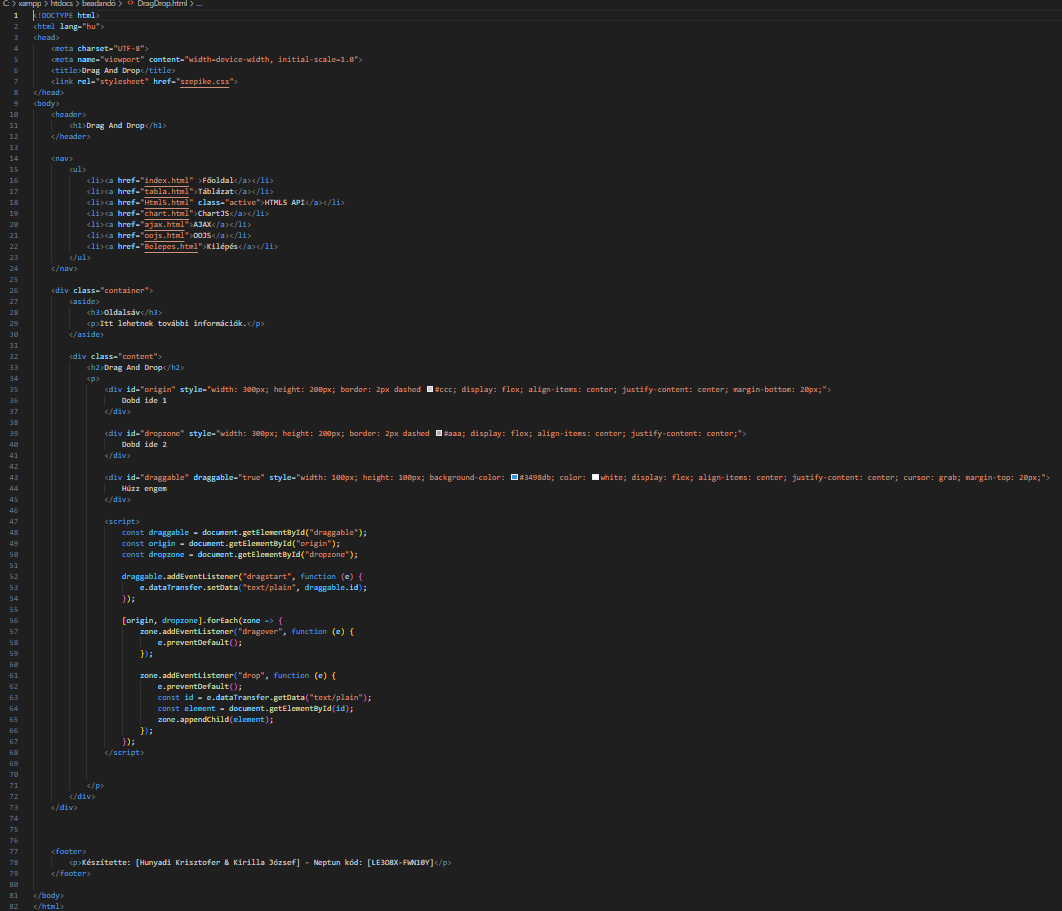
**Kód:**

****

A feladat megpróbálja lekérni, hol vagyunk most.Ha sikerül, kirajzol egy OpenStreetMap térképet, és megjelöli a pozíciónkat egy jelölővel.Ha nem engedjük meg, kiírja az (hiba)üzenetet.A térképet a Leaflet.js könyvtár jeleníti meg.

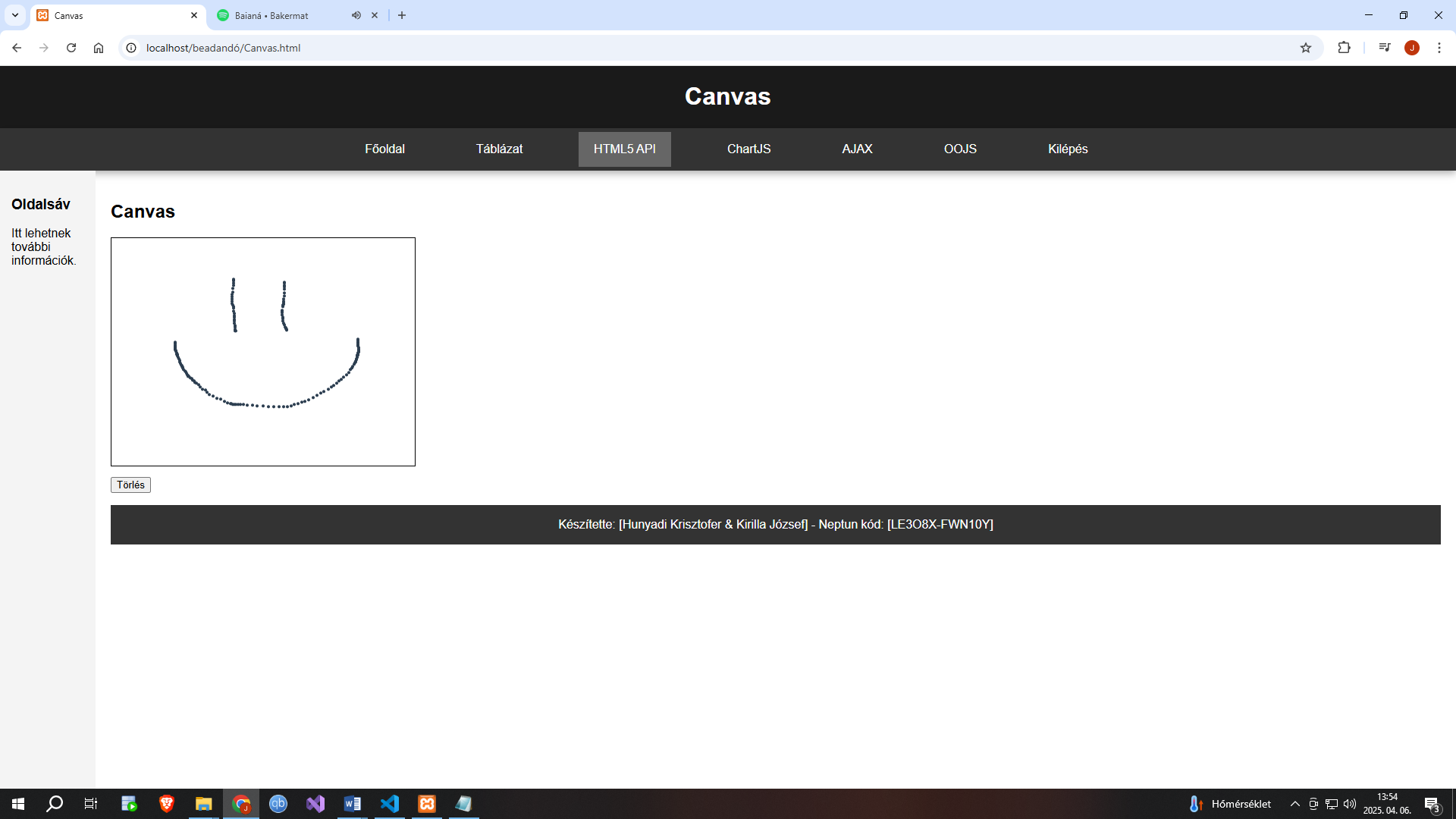
**Drag and drop API**



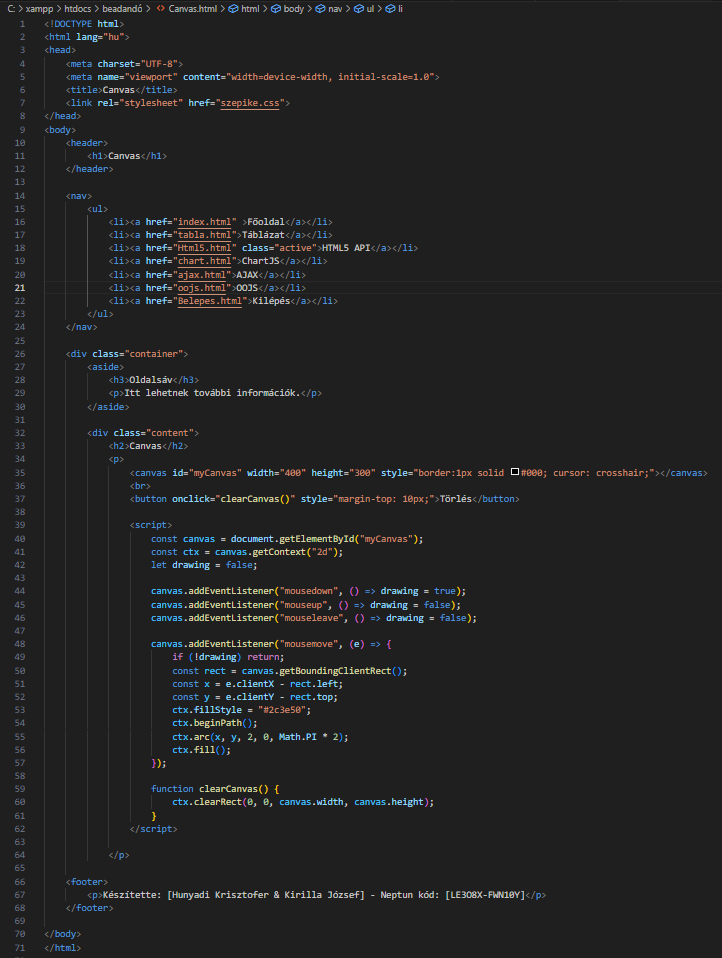
****

Van egy kék doboz, amit húzni lehet. Két zóna van: „Dobd ide 1” és „Dobd ide 2”. A dobozt bármelyik zónába át lehet húzni. A JavaScript kezeli a húzás indítását (dragstart), az áthúzást (dragover) és az elengedést (drop).

**Canvas**

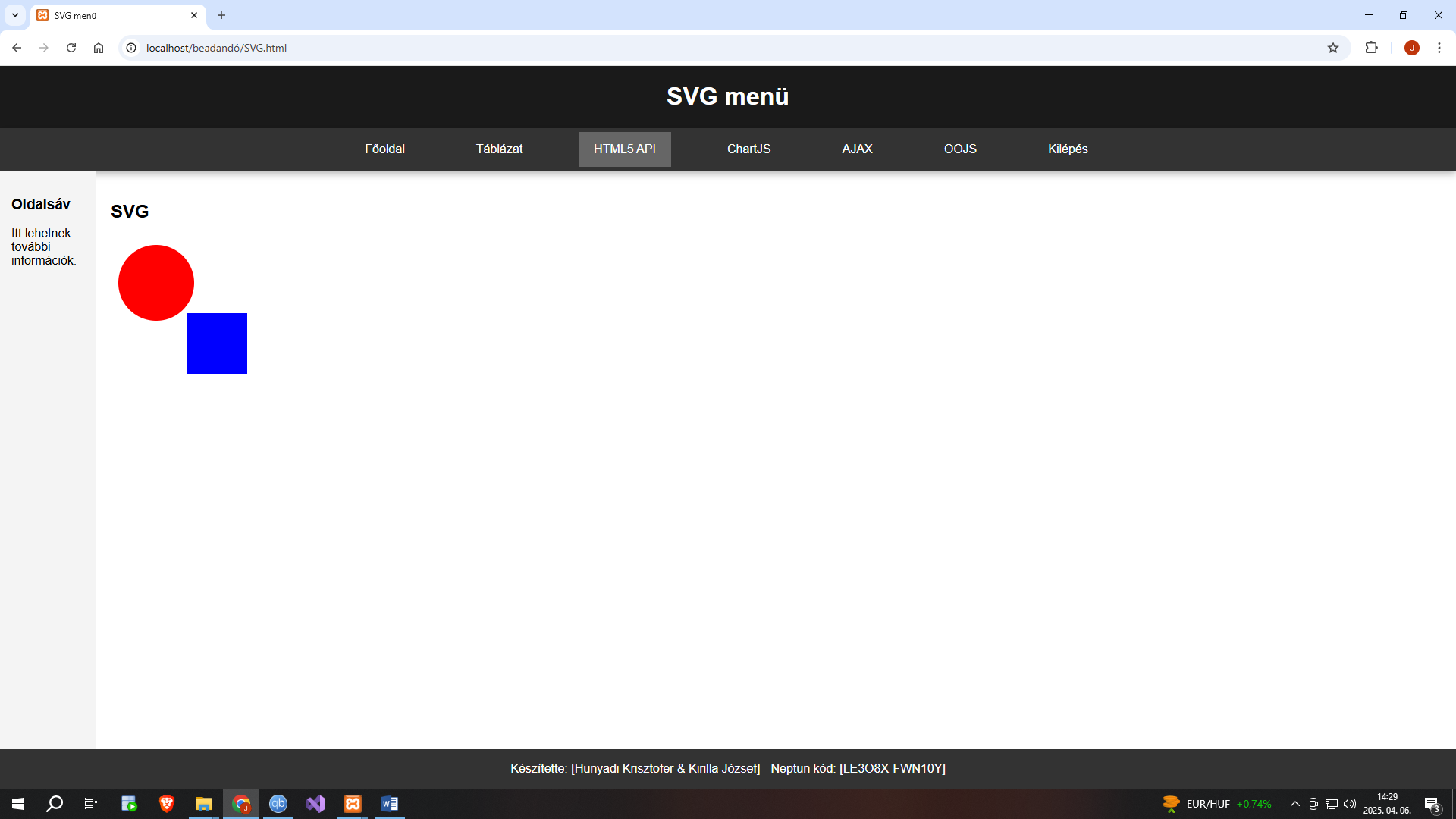


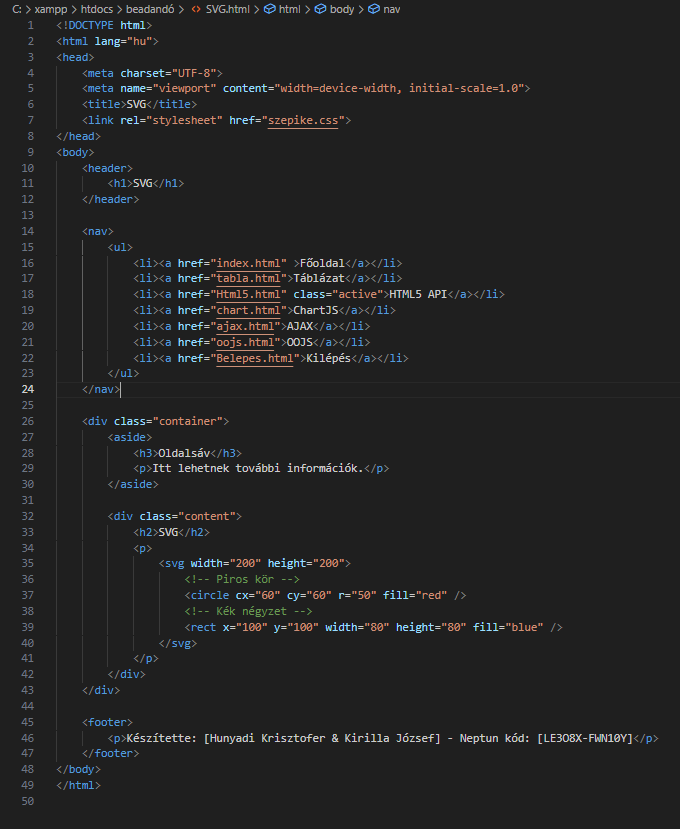
**Canvas kód:**

****

Ez a HTML oldal egy rajzolós weboldalt készít. A középső részen van egy canvas elem, amire egérrel lehet rajzolni.A JavaScript rész figyeli az egér mozgását:ha lenyomva van az egér, akkor kis köröket rajzol a kurzor helyére,ha elengedjük vagy elhúzzuk az egeret a canvasról, leáll a rajzolás.Van egy „Törlés” gomb is, amivel le lehet törölni a teljes rajzot a vászonról.

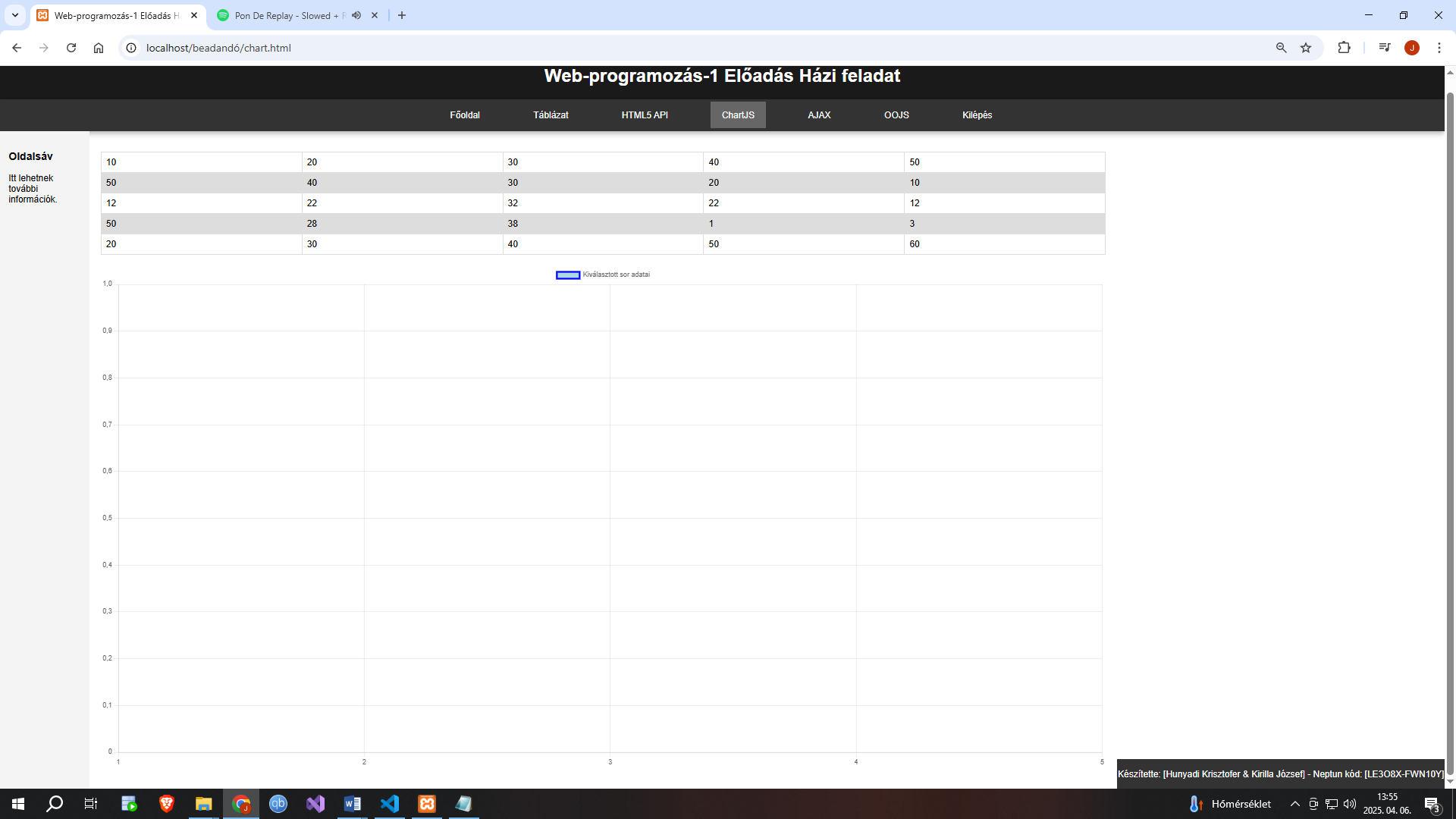
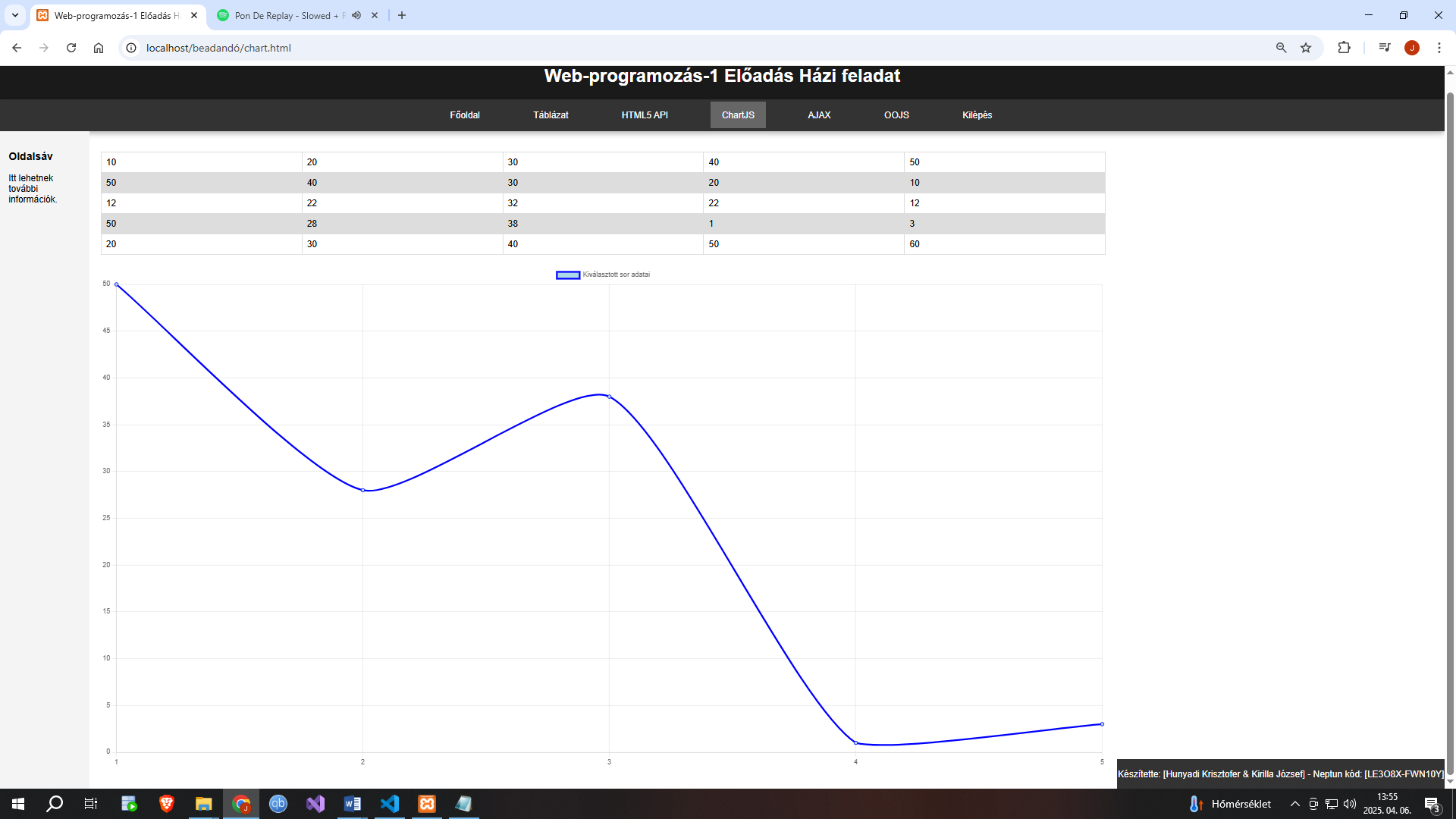
**SVG**



****

Ez a HTML oldal egy egyszerű SVG-s példát mutat be.A tartalmi részben van egy SVG rajz, amiben:egy piros kör van (60,60 ponton, 50-es sugárral),alatta egy kék négyzet (100,100 ponton, 80x80 méretben).

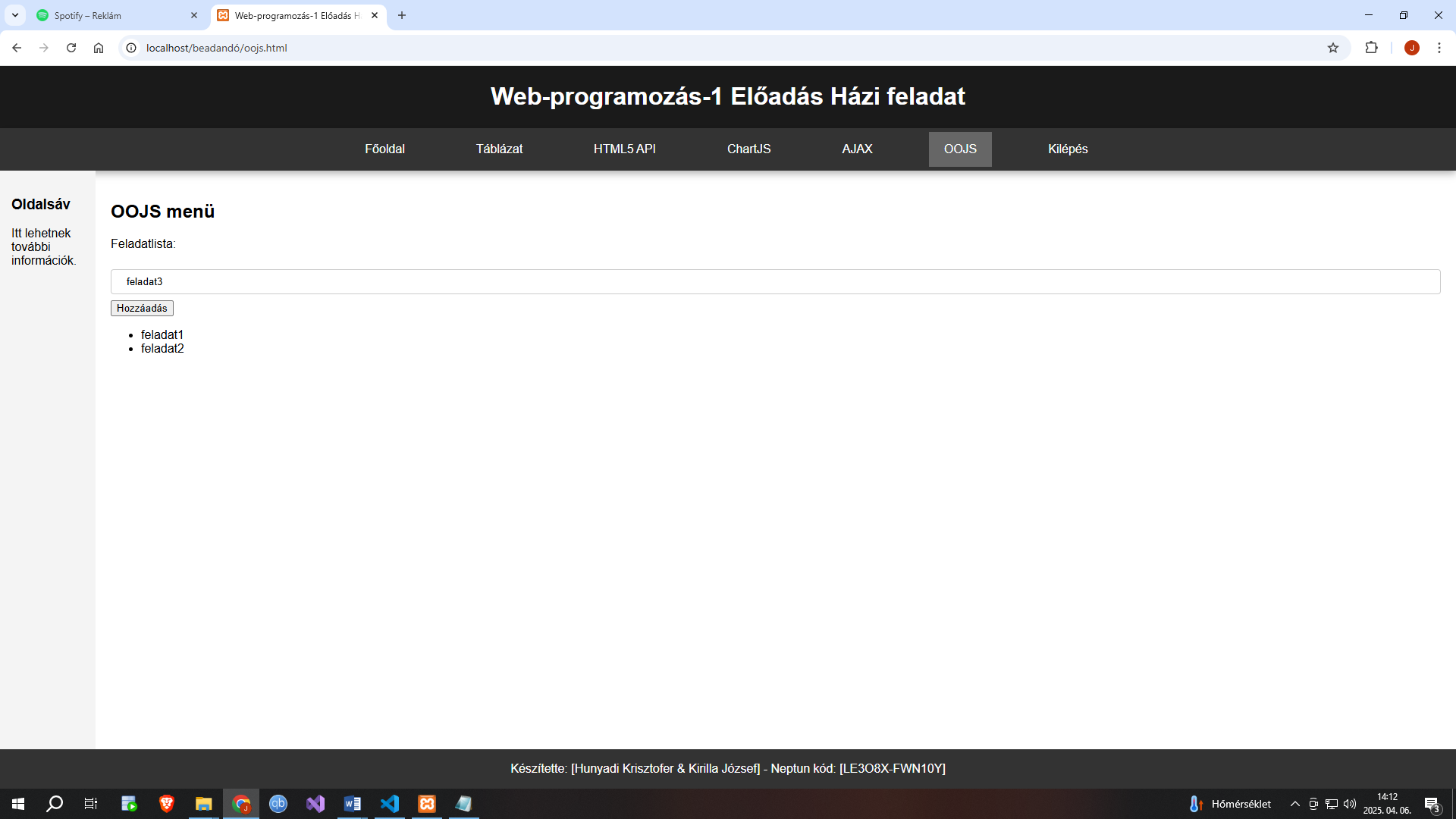
**ChartJS**

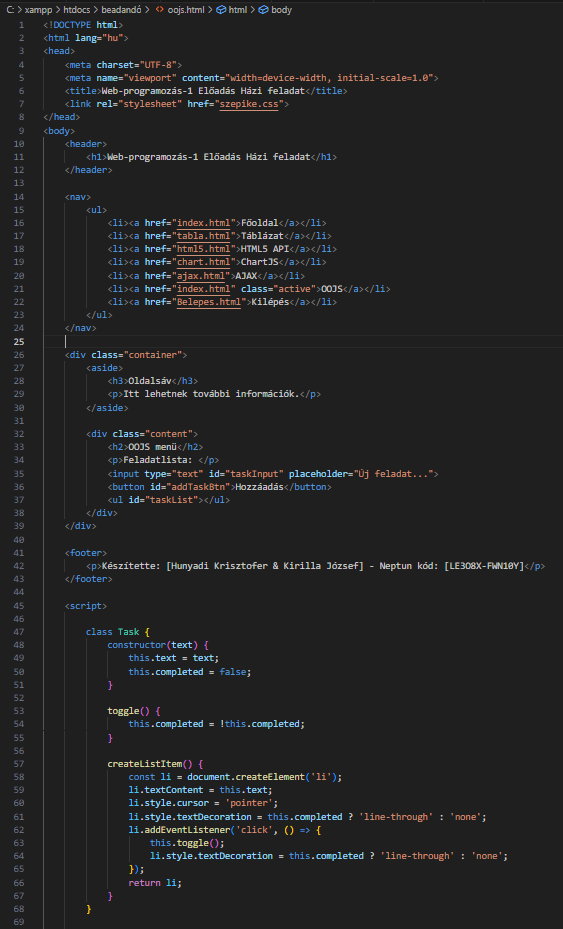
 



Ez a HTML oldal egy interaktív táblázatot és grafikont tartalmaz.A táblázatban számok vannak, és ha rákattintunk egy sorra, akkor a Chart.js segítségével az adott sor adatait megjeleníti egy vonaldiagramon.A JavaScript figyeli, hogy melyik sorra kattintunk, és annak az értékeit rajzolja ki.A Chart.js külső könyvtárat használja a grafikon kirajzolásához.

**OOJS menü**



**** ****

Ez az oldal egy egyszerű feladatlista alkalmazást valósít meg, objektumorientált JavaScript (OOJS) használatával.

A kód három osztályt használ:

Task: egy feladatot tárol, tudja magát áthúzni.

LoggedTask: Task-ból származik, de ki is írja konzolra, ha létrejön.

TaskList: kezeli az összes feladatot és megjeleníti őket a képernyőn.

**Kilépés gomb pedig visszadob a belepes.html- oldalra. Ezzel le is zárnám a dokumentációt. Mivel a végéhez értünk. Köszönjük a figyelmet!**