Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Informatikai Intézet

2025/2026. tanév I. félév

Galaxy weboldal

Web technológiák 1. (GEIAL331-B2)

Jegyzőkönyv

Készítette: Erdélyi Péter (JH3V7T)

Gyakorlatvezető: Dr. Agárdi Anita

Miskolc, 2025. december 07.

TARTALOMJEGYZÉK

1. FELADATLEÍRÁS

Jelen jegyzőkönyv tartalmazza a Web technológiák 1. című tantárgyból készített évközi feladatomat. Ebben a feladatban egy HTML weboldal fejlesztésével foglalkoztam, ahol az oldal formázásához CSS-t, az interaktivitás megvalósításához pedig JavaScript-et vettem igénybe. A Galaxy nevet viselő webhely témaköre a világűrhöz és űrkutatáshoz kapcsolódik, amely az idelátogatók számára ismeretterjesztő tartalmakat kínál.

Az alábbiakban röviden ismertetem a projekt szerkezetét:

- Mappák:
  - `css/` – CSS stílusfájlok
  - `data/` – Adatok betöltésére szánt JSON fájlok
  - `documentation/` – Jegyzőkönyv
  - `images/` – Az oldalon megjelenő fényképek és logók
  - `js/` – JavaScript programok
  - `partials/` – Részleges HTML fájlok (fejléc és lábléc)
- Fájlok:
  - `articles.html` – Blogbejegyzések gyűjteménye érdekes témákról
  - `contact.html` – Kapcsolatfelvételi űrlap
  - `gallery.html` – Különféle kategóriájú képek galériája
  - `index.html` – Kezdőlap
  - `planets.html` – A Naprendszer bolygóinak bemutatása
  - `timeline.html` – Az űrkutatás főbb eseményeinek idővonalas ábrázolása

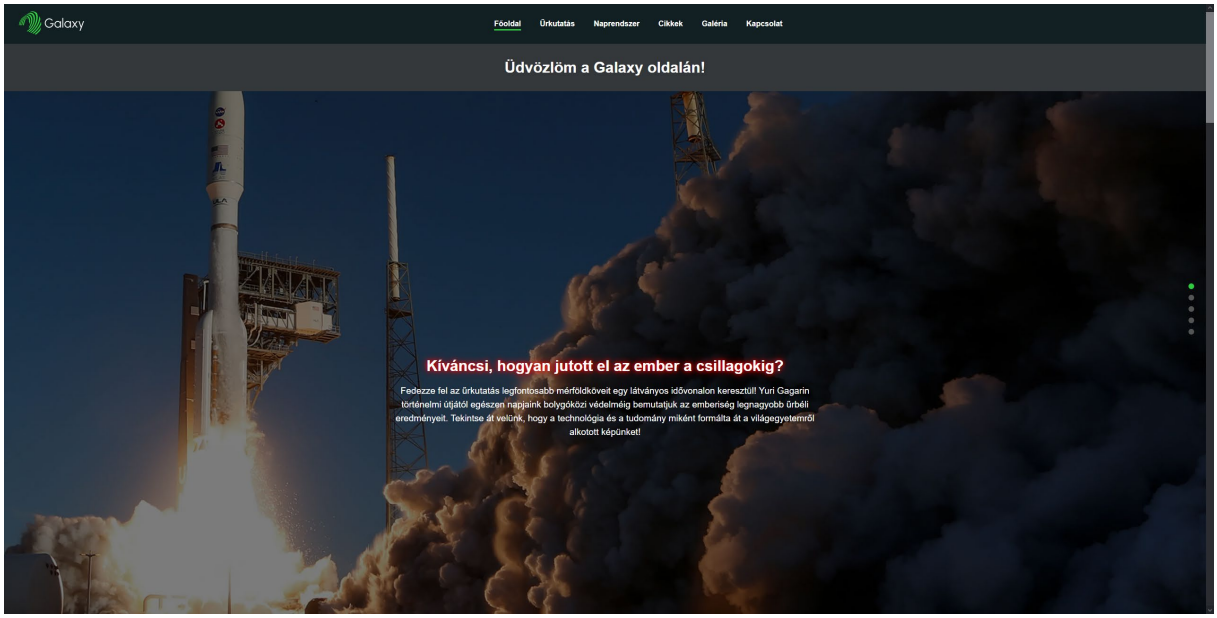
A jegyzőkönyv hátralévő részében bemutatom a Galaxy weboldal felhasználói felületét, valamint elemzem a formázásra és szkriptekre vonatkozó kódot is.

2. KEZDŐLAP

Az `index.html` a Galaxy kezdőlapját jelképezi, melynek célja a felhasználó érdeklődését felkelteni, a webhely főbb funkcióiról egy gyors áttekintést adni.

2.1. A kezdőlap felépítése

A kezdőlap egy részét az 1. ábra szemlélteti.



1. ábra: Kezdőlap felülete

Az `index.html` fájlt `<section>` elemekre osztottam fel, amik tulajdonképpen az 1. ábrán is látható nagy zónákban lévő tartalmat ölelik fel. A zónák az olvasónak nyújtanak útmutatást az oldal menüpontjaival kapcsolatban, és mivel a képernyő teljes szélességét és magasságát lefedik, ezért mindegyikhez külön háttérképet állítottam be. Ezen háttérképek és a szövegek együttes kezelése – különös tekintettel az olvashatóságra – okozott némi nehézséget. A 2. ábra egy viszonylag elegáns megoldást vázol fel erre a problémára.

```css
.zone::before {
    content: "";
    width: 100%;
    height: 100%;
    position: absolute;
    top: 0;
    left: 0;
    background-image: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5));
    z-index: 1;
}
```

2. ábra: A háttérképek és a szövegek szétválasztása

A gond abból származik, hogy ha a teljes zónára alkalmazok egy háttérképet, akkor az a benne lévő szöveget nehezen olvashatóvá teszi. A CSS `opacity` értékének változtatása nem segít, mert az nemcsak a háttérképre, hanem a szövegre is hatással van. Éppen ezért a háttérképek mellett egy egyszerű fekete színátmenetet használtam, aminek 50%-os átlátszóságot adtam, így lényegében a háttérképek fényereje a felére esik vissza.

Fontos kiemelni, hogy ez a színátmenet a `.zone` osztályban mindig felülírásra kerül (mert ugyanúgy a `background-image` része), ha az egyes háttérképek tallózásakor nem definiálunk színátmenetet is. Ennek kiküszöbölése érdekében a legjobb eredményt a `::before` szelektorral és a `z-index` kombinációjával értem el. A szelektor lehetővé teszi, hogy a zónák elé illesszem be a háttérképet, amit aztán a z-index következtében egy másik rétegben helyezek el. Ennek a rétegnek a szövegnél alacsonyabb prioritást adtam, ezáltal a szöveg mögött jelenik meg és különválik tőle.

2.2. Fejléc és lábléc megjelenítése

A fejlécet és láblécet dinamikusan töltöm be JavaScript segítségével nemcsak a kezdőlapon, hanem az összes HTML fájlban. Ennek a módszernek a legnagyobb előnye, hogy módosítás esetén csak egy helyen szükséges a változtatásokat eszközölni. HTML-ben a beillesztés helyén egy placeholder elemet helyezek el, amit az `insertAdjacentHTML()` metódus a megfelelő kódra cserél. Itt a szintaktikai hibák elkerüléséhez csak „részleges” HTML-t (ami pl. nem tartalmaz `<body>` és `<head>` részeket) célszerű beszúrni, erre mutat példát a 3. ábra.

partials > header.html > ...

```
<nav class="navbar">
    <div class="logo">
        <img src="images/logo.png" alt="Logó" height="50px">
    </div>
    <ul class="navbar-links">
        <li><a href="index.html">Főoldal</a></li>
        <li><a href="timeline.html">Űrkutatás</a></li>
        <li><a href="planets.html">Naprendszer</a></li>
        <li><a href="articles.html">Cikkek</a></li>
        <li><a href="gallery.html">Galéria</a></li>
        <li><a href="contact.html">Kapcsolat</a></li>
    </ul>
</nav>
```

3. ábra: Részleges HTML (fejléc)

A fejlécben találhatók a Galaxy főbb navigációs pontjai, melyek kattintás hatására elirányítják a felhasználót az adott oldalra. A navigáció linkjeit egy listában tárolom, a listaelemes megjelenítést viszont a `list-style: none` CSS tulajdonsággal elrejtem.

2.3. Interaktív navigáció

A kezdőlap jobb oldalán kialakítottam egy navigációs segédmenüt, amellyel lehetőség nyílik a zónák közötti gyors váltásra. A zónákat jelképező pontok folyamatosan figyelik azt, hogy a weboldalnak éppen melyik szegmensében tartózkodunk: ennek megvalósítása az Intersection Observer API-val történik (lásd 4. ábra). Ha egy zóna a nézőtérben láthatóvá válik, akkor a `classList` tulajdonságot felhasználva az adott pont elemhez hozzáadok egy új osztályt (`visible`), ennek segítségével pedig a jelenleg aktív pont kinézetét CSS-el tudom módosítani.

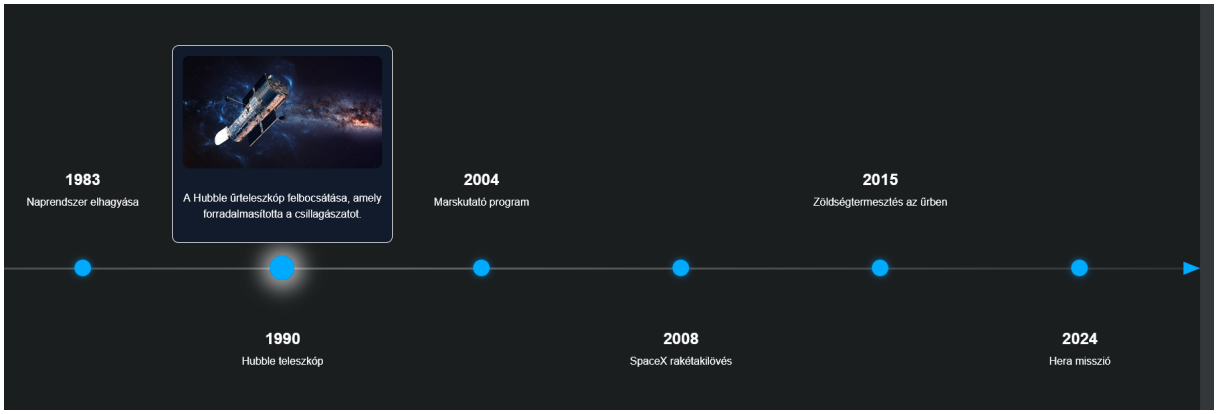
```
const observer = new IntersectionObserver(entries => {
    entries.forEach(entry => {
        if (entry.isIntersecting) {
            entry.target.classList.add("visible");

            const id = entry.target.id;
            dots.forEach(dot =>
                dot.classList.toggle("active", dot.dataset.target === id)
            );
        }
    });
}, { threshold: 0.6 });
```

4. ábra: Az Intersection Observer API használata

További interaktív hatásokat a jQuery könyvtárral hoztam létre. Görgetésre megjelenik egy gomb, amivel a látogató bármikor visszaugorhat az oldal tetejére. A `fadeIn()` és `fadeOut()` metódusokkal a gomb láthatósága fokozatosan változik, emellett az `animate()` egy simább görgetést biztosít.

3. ŰRKUTATÁS

A `timeline.html` az űrkutatás főbb eseményeit hivatott ábrázolni egy interaktív idővonalon. Az idővonal egy részletét mutatja az 5. ábra.



5. ábra: Az űrkutatás idővonala (részlet)

Az eseményekhez kötődő HTML elemeket az `event` osztály foglalja egységbe. Kezdetben csak az évszámok és főcímek olvashatók az idővonalon, hover effektus hatására azonban láthatóvá teszem az `event-details` ablakot, amely már egy fényképet és rövid leírást is csatol egy-egy eseményhez. Magukat az `event` elemeket relatív és abszolút pozícionálási technikákat ötvözve rajzolom ki.

Az idővonal további sajátossága még, hogy az információs blokkokat felváltva helyeztem el. Ez kivitelezhető az `:nth-child()` pszeudo-osztállyal (lásd 6. ábra) tisztán CSS formában, tehát a váltakozó megjelenítéshez nincs szükség JavaScript-re.

```css
.event:nth-child(odd) .info {
    bottom: 100px;
}

.event:nth-child(even) .info {
    top: 100px;
}

.event:nth-child(odd) .event-details {
    top: 50px;
    bottom: auto;
}

.event:nth-child(even) .event-details {
    top: auto;
    bottom: 50px;
}
```

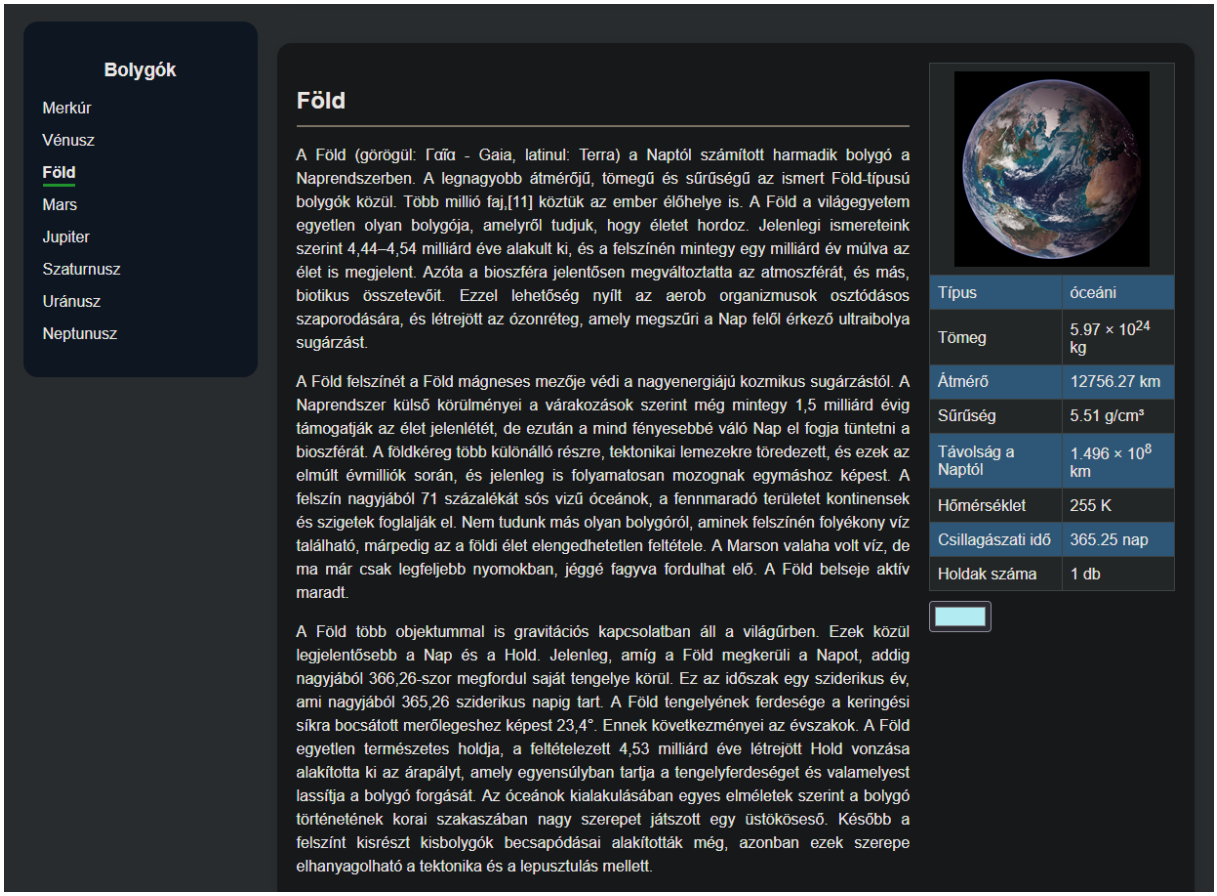
6. ábra: Váltakozó megjelenítés

4. NAPRENDSZER

A `planets.html` a Naprendszerünk bolygóit mutatja be közelebbről. Ezt az oldalt a következő főbb részek alkotják:

- navigációs oldalsáv
- minden bolygó esetén:
  - szöveges leírás
  - adatok táblázatos formátumban
  - fénykép

A 7. ábra a Naprendszer menüpont felületét illusztrálja.



7. ábra: A Naprendszer oldal Föld bolygója

4.1. Dinamikus tartalomgenerálás

A `planets.html` tartalmát dinamikusan generálom a `planets.json` fájlból. Ehhez egy AJAX kérést hajtok végre a JavaScript `fetch()` metódusán keresztül.

4.1.1. Táblázatok előállítása

A táblázatok előállításának logikájáért a `createTable()` függvény felel. A 8. ábrán látható `for` ciklus elvégzi a bolygó számszerű adatainak beillesztését.

```
for (const [key, value] of Object.entries(planet.data)) {
    const tableDatakey = document.createElement('td');
    tableDatakey.textContent = dataMap[key] || key;
    let displayValue = value;

    if (key === 'mass' || key === 'distance') {
        const exp = value.toExponential();
        const [mantissa, exponent] = exp.split('e');
        displayValue = `${mantissa} × 10<sup>${Number(exponent)}</sup>`;
    }

    const tableRow = document.createElement('tr');
    const tableData = document.createElement('td');
    tableData.innerHTML = displayValue + (unitMap[key] ? ' ' + unitMap[key] : '');

    tableRow.appendChild(tableDatakey);
    tableRow.appendChild(tableData);
    table.appendChild(tableRow);
}
```

8. ábra: A táblázatok feltöltése adatokkal

A `document.createElement()` metódussal különböző HTML elemeket lehet dinamikusan létrehozni, beleértve a táblázatkezeléshez szükséges `<tr>`, `<td>` stb. tagokat is. A `dataMap[]` és `unitMap[]` tömbök a kulcs-érték párokat tárolják, melyek a JSON fájl mezői és a táblázatban szereplő információk közötti tetszőleges leképezést teszik lehetővé. Amint a HTML elemek feltöltésre kerültek adattal, úgy ezekre külön-külön meghívom az `appendChild()` metódust, ami gondoskodik a szülőelemhez való hozzáfűzésről.

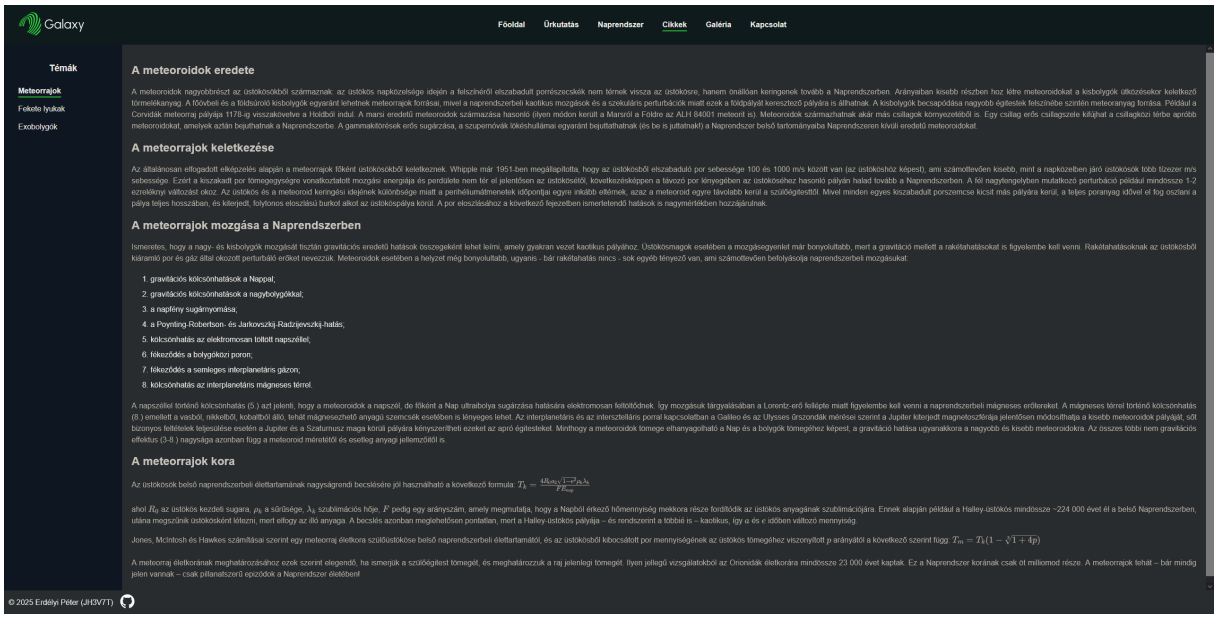
4.1.2. Szöveg és oldalsáv előállítása

Ugyanígy működik a Naprendszer oldalon lévő többi elem generálása is. A JSON-ben található bolygókat iteratívan be kell járni, majd a megfelelő HTML elemeket az előbbi fejezetben ismertetett metódusokkal kell létrehozni és beszúrni a dokumentumba. A címsorokat pl. a bolygók neve alapján alakítom ki, míg a bekezdéseket a `description` mezőben az elválasztókarakterek (vesszők) határánál zárom le.

A `planets.html` oldalsávja könyvjelzőként funkcionál, amely a kezdőlaphoz hasonló módon segíti elő a bolygók közötti navigációt. CSS formázáshoz beállítottam a `position: sticky` szabályt, melynek eredményeképpen az oldalsáv követi az ablakon belüli görgetést, így a navigációs menü folyamatosan látható marad.

5. CIKKEK

Az `articles.html` fájlban olyan cikkek vagy blogbejegyzések találhatók, amik a világűrrel kapcsolatos témákat dolgoznak fel. A cikkek oldalra lépve az alábbi (9. ábra) látvány tárul elénk:



9. ábra: A cikkek menüpont tartalma

Az itteni szöveg formázására alkalmaztam a HTML beépített szövegformázási eszközeit: pl. egyes szövegrészek dőlt betűs megjelenítéshez `<i>` tagokat helyeztem el, vagy egy bekezdésen belül sortörést iktattam be a `<br>` címke használatával.

A cikkek menüpontban a felhasználó pillanatnyilag 3 különböző téma közül választhat, amelyek az előzőekben megszokott módon egy oldalsó navigációs sávban vannak feltüntetve. A következő fejezetek néhány érdekesebb HTML elem formázását vázolják fel.

5.1. Képletek megjelenítése

A képletek szerkesztéséhez a külsős KaTeX könyvtárat vettem igénybe, amely biztosítja a LaTeX formátumban begépelt formulák helyes megjelenítését. A KaTeX képes az inline módon megadott képleteket kirajzolni, ha azokat szintaktikailag jól kezeljük, vagyis nyitó `\(` és záró `\)` tagok közé tesszük. A könyvtárhoz létezik egy `auto-render.min.js` szkript is, amely gondoskodik az automatikus renderelésről, ha a `<script>` címkén belül az `onLoad` attribútum értéket a következőképpen állítjuk be:

```
onload="renderMathInElement(document.body);"
```

5.2. Képaláírások elhelyezése

Az ábrák alatt lévő képaláírást a `<figure>` HTML elem felhasználásával valósítottam meg. Egy ilyen `<figure>` elemet demonstrál a 10. ábra.

```
<figure>
    <img src="images/articles/m87_black_hole.png" alt="M87 fekete lyuk" width="800px">
    <figcaption>A Messier 87 galaxis közepében található fekete lyuk</figcaption>
</figure>
```

10. ábra: A `<figure>` HTML elem működése

A képaláírást a `<figcaption>` részben kell megadni, amit természetesen CSS-el igény szerint testre lehet szabni. Annak érdekében, hogy a magyarázószöveg vizuálisan elkülönüljön a cikk többi tartalmától, a `<figcaption>` egy teljesen eltérő betűtípust alkalmaz, és dőlt betűsen jelenik meg.

5.3. Idézetek beillesztése

Hosszabb idézetek beillesztése HTML-ben a `<blockquote>` címkével történik, illetve rövid gondolatok közlésére használható még a `<q>` elem is. Előbbinél a böngésző automatikus behúzást kezdeményez, az utóbbinál pedig a tartalmat idézőjelek közé teszi. A Galaxy oldalon mindkét esetre található példa, melyek közül a `<blockquote>` elemet CSS formázással emeltem ki. Itt az idézet mellett a `<footer>` részbe beírtam magát az eredeti szerzőt is, amely `<blockquote>` tagoknál alkalmas a forrás megjelölésére.

5.4. Videó beágyazása

Az „Exobolygók” című blogbejegyzésbe beágyaztam egy YouTube videót, amihez szükséges volt a YouTube saját IFrame API-ját beemelni a `<script>`-ek közé. Az IFrame API nélkülözhetetlen az ilyen videók JavaScript-el való vezérléséhez, így pl. a videóban történő ugrás implementálásához is.

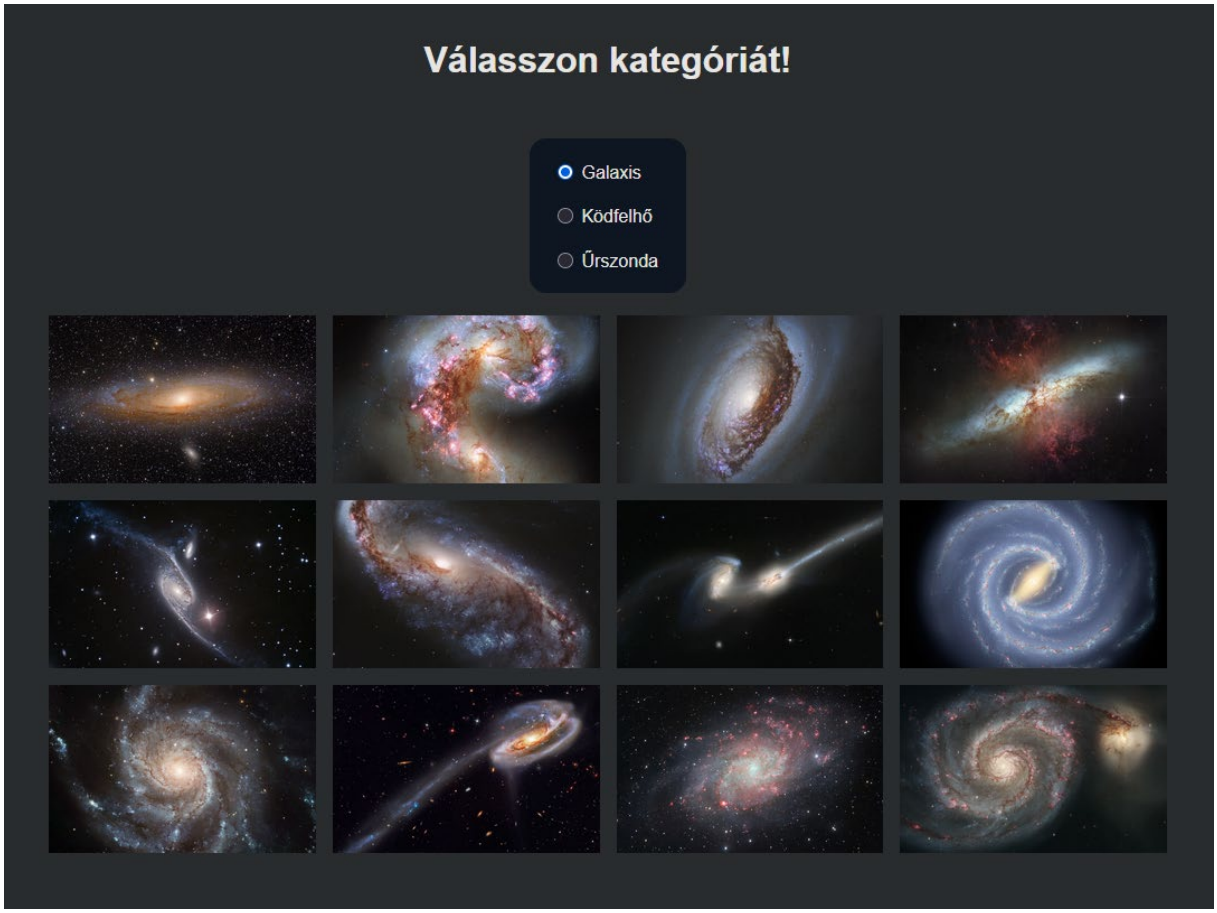
Az `onYouTubeIFrameAPIReady()` egy olyan függvény, ami az API készenléti állapotának elérésekor végrehajtásra kerül. Ebben a függvényben definiálni kell a videó paramétereit, többek között az azonosítót, lejátszó méreteit, valamint a hozzárendelt eseményeket is. A feladatban a `seek-button` gombhoz egy eseményfigyelőt adtam hozzá, amely kattintás hatására a videóban a 65. másodperchez ugrik. A beágyazás menete a 11. ábrán követhető nyomon.

```
var videoPlayer;

function onYouTubeIframeAPIReady() {
    videoPlayer = new YT.Player('video-player', {
        videoId: 'yv4DbU1CWAY',
        height: '450',
        width: '800',
        events: {
            'onReady': onPlayerReady
        }
    });
}

function onPlayerReady(event) {
    document.getElementById('seek-button').addEventListener('click', function() {
        videoPlayer.seekTo(65);
    });
}
```

11. ábra: YouTube videó beágyazása és JavaScript-el történő vezérlése

6. GALÉRIA

A `gallery.html` egy interaktív galériát foglal magában (lásd 12. ábra). A képeket rácsos (`grid`) elrendezésben jelenítem meg, melynek CSS kódrészét a 13. ábrán mutatom be.



12. ábra: A galéria felhasználói felülete

```css
.gallery {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    grid-template-rows: repeat(3, 1fr);
    grid-column-gap: 15px;
    grid-row-gap: 15px;
    width: 50%;
    max-width: 1000px;
    margin: 0 auto;
}
```

13. ábra: A galéria rácsos elrendezésének CSS beállításai

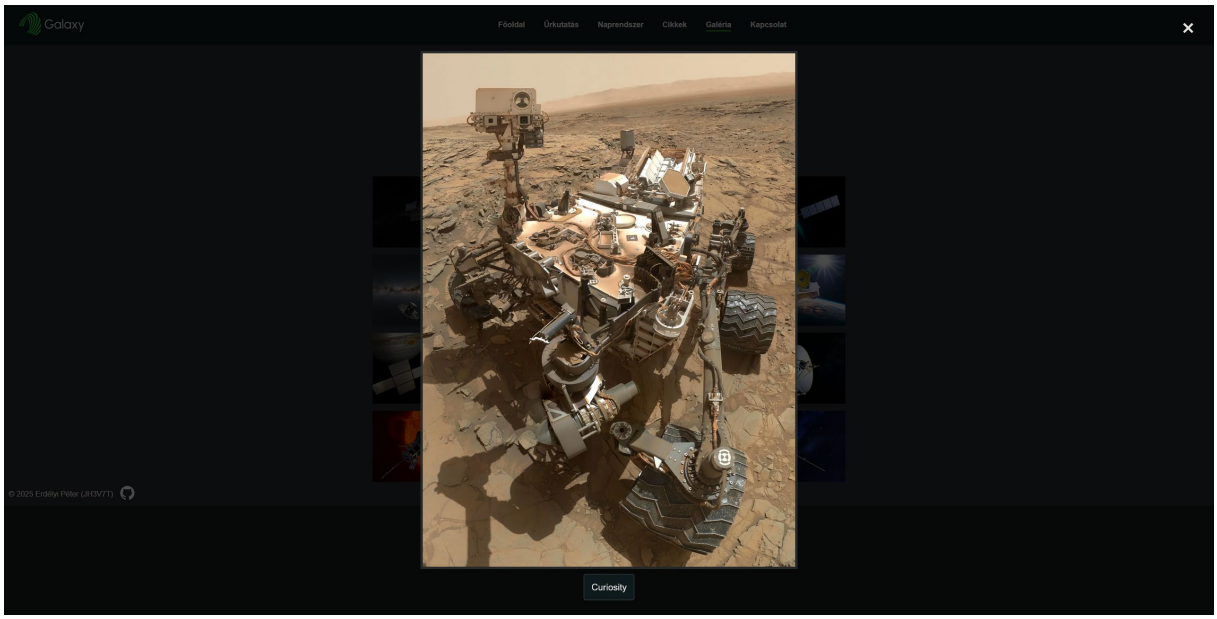
6.1. Fényképek betöltése

A galériát nézegető személy egy rádiógomb segítségével ki tudja választani, hogy milyen kategóriájú fényképeket szeretne megtekinteni. Amikor a kategóriát átállítja, akkor lefut a `loadGallery()` függvény kódja, amely a galériában látható képeket dinamikusan frissíti egy `fetch()` metódussal, ehhez a szükséges adatokat pedig egy JSON fájlból olvassa ki.

A Galaxy weboldal galériájánál alapvető szempont volt az, hogy a fényképeket webes megtekintésre optimalizáljam, mert a lassú betöltési sebesség ronthatja a felhasználói élményt. Ehhez a képek többségét webp formátumba konvertáltam át, amely képes a fájlméretet elfogadható minőségben, hatékonyan minimalizálni. Ennek köszönhetően kategóriaváltáskor a betöltés folyamata gyorsan lezajlik, így ez a felhasználó számára nem okoz fennakadást.

6.2. Fényképek kinagyítása

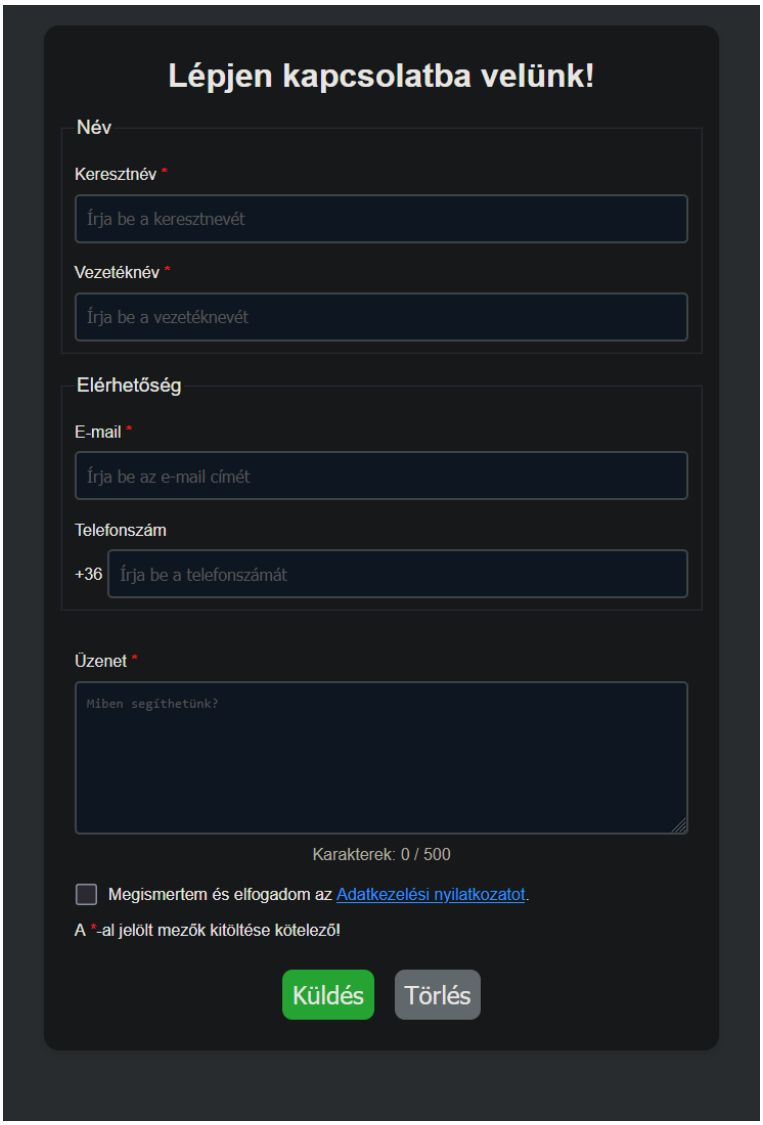
Bármelyik fénykép kinagyítható kattintással, amit egy ún. lightbox technikával (lásd 14. ábra) oldottam meg. Ennek lényege, hogy a kinagyított kép egy felugró ablakban jelenik meg, ami a háttérben lévő többi tartalmat teljesen elfedi azáltal, hogy a legfelső rétegben helyezkedik el (a z-index értéke rendkívül magas). A lightbox megnyitásakor olvasható lesz az adott fénykép leírása, a bezáráshoz pedig elég csak a képen kívüli területre kattintani. Ha ez utóbbi valakinek nem tűnik fel, akkor a bezárást „hagyományos” módon is megteheti a lightbox jobb felső sarkában lévő X-re nyomva.



14. ábra: Lightbox megjelenítés

7. KAPCSOLAT

A `contact.html` gyakorlatilag egy kapcsolatfelvételi űrlapként szolgál, ahol a látogató közvetlen kapcsolatba léphet a Galaxy oldal szerkesztőivel. Az űrlap kinézetét a 15. ábra demonstrálja.



15. ábra: A kapcsolatfelvétel űrlapja

7.1. Alkalmazott űrlapelemek

A kapcsolati menüpont oldalának legfőbb része a `<form>` elem. Ez egy olyan tároló, amelyben a felhasználóktól érkező inputokat gyűjtjük, különböző `<input>` címkék felhasználásával. A kapcsolatfelvétel HTML szerkezetének egy részlete figyelhető meg a 16. ábrán.

```
<form>
    <h1>Lépjen kapcsolatba velünk!</h1>

    <fieldset>
        <legend>Név</legend>

        <label for="first-name">Keresztnév <span class="required-star">*</span></label>
        <input type="text" name="first-name" id="first-name" class="name" pattern="^[\p{L}\s]+$" title="Csak betűk és szóközök lehetségesek"

        <label for="last-name">Vezetéknév <span class="required-star">*</span></label>
        <input type="text" name="last-name" id="last-name" class="name" pattern="^[\p{L}\s]+$" title="Csak betűk és szóközök lehetségesek" p

        <div class="warning" id="name-warning" role="alert">A név csak betűket és szóközöket tartalmazhat!</div>
    </fieldset>

    <fieldset>
        <legend>Elérhetőség</legend>

        <label for="email">E-mail <span class="required-star">*</span></label>
        <input type="email" name="email" id="email" placeholder="Írja be az e-mail címét" required>
        <ul id="email-list">
            <li>@gmail.com</li>
            <li>@yahoo.com</li>
            <li>@outlook.com</li>
            <li>@hotmail.com</li>
            <li>@freemail.hu</li>
        </ul>
        <div class="warning" id="email-warning" role="alert">Az e-mail cím formátuma helytelen!</div>

        <label for="phone-number">Telefonszám</label>
        <div class="phone-container">
          <span style="margin-right: 5px;">+36</span>
          <input type="tel" name="phone-number" id="phone-number" placeholder="Írja be a telefonszámát" style="align-items: center;">
        </div>
        <div class="warning" id="phone-warning" role="alert">A telefonszám formátuma helytelen!</div>
    </fieldset>
```

16. ábra: A `contact.html` szerkezete (részlet)

A feladatban a következő űrlapelemeket alkalmaztam (a teljesség igénye nélkül):

- `<fieldset>`
- `<label>`
- `<input>`
- `<textarea>`
- `<button>`

A `<fieldset>` pl. az összetartozó űrlapelemek csoportosítására használatos, így a vezeték- és keresztnév, illetve az e-mail cím és telefonszám mezőket külön-külön kategóriákba soroltam. Ezeknek a kategóriáknak a `<legend>` címkével adtam nevet. A kötelezően kitöltendő mezőket egy piros csillaggal jelöltem, amiket a `<span>` elemek következtében inline tudtam formázni. Az inputmezőknél továbbá megadtam placeholder szövegeket is: ezek csak üres mezőkben láthatók, vagyis gépelés után eltűnnek.

7.2. Az űrlap validálása

Az űrlap validálását részben a HTML, részben a JavaScript program végzi, amely folyamatot az alábbiakban részletesen kifejtem.

- A `required` attribútum beállításával a böngésző addig nem engedi az űrlap elküldését, amíg a felhasználó minden ilyen mezőt nem tölt ki.
- A név mezőkhöz megadtam egy `pattern` értéket, amely csak betűket és szóközöket engedélyez. Ha a felhasználó ezt nem tartja be, akkor a `title` kiegészítő szöveg figyelmezteti őt erre, és ekkor a `name-warning` is láthatóvá válik. Mindemellett az inputmezők keretének színe pirosra változik (ez egyaránt teljesül a többi űrlapelemnél).
- Az e-mail cím ellenőrzésére definiáltam egy regex kifejezést, amely alkalmas az alapvetően hibás formátumok kiszűrésére.
- A telefonszám megadása ugyan nem kötelező, de ha a felhasználó ezt a mezőt mégis kitölti, akkor szükség van az itt szereplő érték vizsgálatára. A JavaScript kódot úgy írtam meg, hogy a telefon inputmezője minden nem-számjegy karaktert automatikusan eltávolít, legfeljebb 9 számjegyet enged, ezenfelül a megadott telefonszámot magyar formátumra alakítja.
- Az üzenet hossza nem haladhatja meg az 500 karaktert. Mivel ez egy `<textarea>` elem, ezért a `max-length` attribútum értékét szintén 500-ra állítottam be, emiatt a böngésző letiltja az ennél hosszabb üzenetek írását.
- A felhasználónak el kell fogadnia az Adatkezelési nyilatkozatot, vagyis ki kell pipálnia a jelölőnégyzetet.
- A „Küldés” gomb funkcióját alapértelmezetten kikapcsoltam. Maga a `contact.js` azt figyeli, hogy az e-mail cím és telefonszám formátumok megfelelnek-e az adott regex-nek. Hiba esetén erről egy értesítést küldök az `alert()` metódussal, majd a `focus()` a hibás mezőre irányítja a felhasználót.

Amennyiben a felhasználó sikeresen átmegy a böngésző beépített HTML ellenőrzésén, és emellett egyidejűleg teljesíti a JavaScript-ben található feltételeket is, úgy lefut a `form.submit()` metódus, a beírt adatok elküldésre kerülnek.