URL: <https://prowebsitebuilder.online/eloadas>

GIT: https://github.com/FeketGergely/WebprogBeadando

Index.html: A lang attribútum „hu”-ra van állítva, ami a magyar nyelvű tartalomra utal. A meta charset="UTF-8" biztosítja az ékezetes karakterek helyes megjelenítését, míg a viewport beállítás felel a reszponzív megjelenésért mobil eszközökön.

A head szekcióban található egy beágyazott CSS blokk, amely meghatározza a weboldal megjelenését. A háttér világosszürke (#f4f4f4), a szöveg betűtípusa az Arial családba tartozik. A fejléc (header) zöld háttérrel és fehér betűszínnel jelenik meg, középre igazított címmel. A navigációs sáv (nav) sötétszürke alapszínű, és vízszintesen elrendezett menüpontokat tartalmaz. Az aktív vagy egérrel megközelített menüpont zöld kiemelést kap, így jól látható, melyik aloldal van éppen betöltve.

A fő tartalom a #content azonosítójú <div> elemben található. Ez jelen esetben egy üdvözlő szöveget tartalmaz, amely arra ösztönzi a felhasználót, hogy válasszon a fenti menüpontok közül. A lábléc (footer) szintén sötét alapszínű, középre igazított fehér szöveget tartalmaz, amely az oldal készítőjére és Neptun kódjára vonatkozó információkat jeleníti meg. A lábléc fix pozíciójú, az oldal alján marad görgetés közben is.

A navigációs menü több fájlra hivatkozik: index.html, table.html, html5.html, chartjs.html, ajax.html, oojs.html, react.html. Ezek valószínűleg az egyes tananyagokhoz vagy feladatokhoz tartozó különálló oldalak. Az index.html az alapértelmezett kezdőlap, amit az active osztály jelöl is. A képen szöveg, képernyőkép, szoftver, Weblap látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

Table.html: Ez a HTML dokumentum egy CRUD (Create, Read, Update, Delete) táblázat funkcióval ellátott weboldalt ír le, amely a Web-programozás-1 kurzus részeként készült. A célja, hogy lehetőséget adjon adatok megadására, megjelenítésére, szűrésére, rendezésére, szerkesztésére és törlésére egy egyszerű felületen keresztül. Az oldal modern és letisztult felhasználói felületet kínál, és JavaScript segítségével valósítja meg a funkciókat dinamikusan, külső adatbázis vagy háttérrendszer nélkül.

A dokumentum a Táblázat címet viseli, ami a böngésző fülön jelenik meg. A head szekcióban beágyazott CSS stílus található, amely meghatározza a weboldal kinézetét és elrendezését. A teljes oldal világos, halványszürke háttérrel rendelkezik, a szövegek Arial betűtípussal jelennek meg. A fejléc zöld alapszínű, és középre igazított címsort tartalmaz. A navigációs sáv sötétszürke, vízszintes menüpontokat jelenít meg, amelyek közül a "Táblázat" oldal aktívként van megjelölve. Ez az oldal az egyik több aloldal közül, amelyeket külön fájlokban valósítottak meg.

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A fő tartalomblokk egy táblázat köré épül. A táblázatot az oldal tetején egy szövegmező egészíti ki, amely lehetővé teszi a keresést, vagyis a sorok szűrését kulcsszavak alapján. A táblázat oszlopai a következő mezőket tartalmazzák: ID, Név, Kor, Cím, Telefon, valamint egy oszlop a műveletekhez. A fejléc oszlopai kattinthatók, ezekre kattintva az adatok alfabetikus sorrendben rendezhetők JavaScript segítségével.

A képen szöveg, képernyőkép, szoftver, képernyő látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, képernyő látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, képernyő látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, képernyő látható

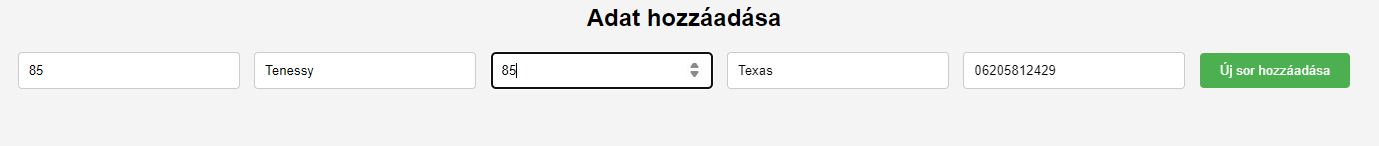
Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A táblázat alatt található egy űrlap, amely új adatok megadását szolgálja. Az űrlap mezői: ID (szám), Név (szöveg, 2–50 karakter), Kor (szám, 1–120 között), Város (szöveg, 2–50 karakter), és Telefon (11 számjegyből álló numerikus szöveg). Az űrlap elküldése után JavaScript validáció ellenőrzi a bevitt adatokat. Ha a mezők nem felelnek meg a megadott feltételeknek, a hibaüzenetek piros színnel jelennek meg az űrlap alatt.

Érvényes adatok esetén a JavaScript dinamikusan létrehoz egy új sort a táblázatban, amely tartalmazza a megadott adatokat, valamint két gombot: „Szerkesztés” és „Törlés”. A „Törlés” gomb eltávolítja az adott sort a táblázatból. A „Szerkesztés” gomb visszatölti a sor adatait az űrlap mezőibe, majd eltávolítja az eredeti sort, így a felhasználó frissítheti az adatokat, és újra hozzáadhatja őket az űrlapon keresztül.

A képen szöveg, képernyőkép, szoftver, Operációs rendszer látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.



A képen szöveg, képernyőkép, szám, Betűtípus látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szám, Betűtípus látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, képernyő látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A JavaScript kód közvetlenül az oldal alján található. Az oldal betöltődése után az eseményfigyelők aktiválódnak: az űrlap beküldése, a rendezhető oszlopfejlécek kattintása és a keresőmező használata mind működésbe lép. Az adatok kezelése teljes mértékben kliensoldalon történik, háttérfolyamat vagy szerveroldali adatmentés nincs jelen. Ez a megoldás oktatási céllal készült, és jól demonstrálja az alapvető DOM-kezelési, eseményfigyelési és adatműveleti technikákat JavaScript-ben.

Html5.html: A fejléc a projekt nevét tartalmazza, míg az alatta elhelyezett navigációs menü segítségével a felhasználó gyorsan elérheti az oldal különböző részeit. A fő tartalom részen belül különálló szekciókban kerülnek bemutatásra a különböző technológiák.

A Web Storage példában a látogató beírhatja a nevét, amely a böngésző helyi tárolójába kerül mentésre. Később ezt az adatot újratöltheti, és egy üdvözlő szöveg jelenik meg a névvel. Ez az interaktivitás bemutatja a localStorage használatának alapjait.

A képen szöveg, képernyőkép, Betűtípus, embléma látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A Web Workers részben egy háttérszálon futó számláló kerül elindításra. A felhasználó egy gomb segítségével elindíthatja a számlálást, majd leállíthatja azt, a számláló értéke ezután megjelenik. Ez a szekció demonstrálja, hogyan lehet JavaScript-ben párhuzamos szálakat indítani a fő szál terhelése nélkül.

A képen szöveg, Betűtípus, képernyőkép, embléma látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.A képen szöveg, Betűtípus, képernyőkép, Grafika látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A Server-Sent Events rész bemutatja, hogyan lehet a szervertől valós idejű adatokat fogadni. Egy gomb megnyomása után a böngésző elkezd kommunikálni egy PHP szkript segítségével, amely rendszeres időközönként küld adatokat az ügyfél felé. Ez az aszinkron technológia alkalmas például értesítések vagy élő frissítések megjelenítésére.

A képen szöveg, Betűtípus, képernyőkép, embléma látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, Betűtípus, képernyőkép, embléma látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A Geolocation API szekció lehetővé teszi a felhasználó földrajzi helyének meghatározását a böngészőn keresztül. Az engedély megadása után a koordináták megjelennek az oldalon. Ez a funkció főként mobil eszközökön vagy térképes alkalmazásokban hasznos.

A képen szöveg, képernyőkép, szoftver, multimédia látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, Betűtípus, képernyőkép, embléma látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A Drag and Drop API példában a látogató egy piros négyzetet húzhat át egy kijelölt dobozba. Az események kezelését JavaScript végzi, és a drag-drop élményt vizuális visszacsatolás egészíti ki.

A képen szöveg, képernyőkép, Betűtípus, Téglalap látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, Betűtípus, Téglalap látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A Canvas rajzolási funkcióval a felhasználó szabadkézi vonalakat rajzolhat az egér segítségével egy vászonra. A mozgást nyomkövető események figyelik, és a vonalakat a 2D rajzolási API rendereli. Egy külön gombbal lehetőség van a vászon tartalmának törlésére is.

A képen szöveg, képernyőkép, Téglalap, képernyő látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen vázlat, diagram, rajz látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

Az SVG és Canvas összehasonlítása céljából egy fix grafikai példa kerül megjelenítésre: egy kör, egy négyzet, egy átlós vonal és egy szöveg. Az SVG és Canvas verziók ugyanazt a képet jelenítik meg két külön technológiával. Egy csúszka segítségével mindkettő nagyítható, amely látványosan szemlélteti a két grafikai API közötti különbségeket, például az SVG vektoros skálázhatóságát és a Canvas bitmap-alapú viselkedését.

A képen képernyőkép, diagram, szöveg, Diagram látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen diagram, képernyőkép, sor, Diagram látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

Chartjs.html/chartjs.js

dinamikus vonaldiagramot jelenít meg a Chart.js könyvtár segítségével. A célja annak bemutatása, hogyan lehet HTML, CSS és JavaScript használatával vizualizálni adatokat egy táblázatból.

Az oldal felépítése egy szabványos HTML5 dokumentumszerkezetet követ. A fejlécrész (<header>) tartalmazza az oldal címét, amely jól láthatóan megjelenik az oldal tetején. Az oldal navigációs sávja (<nav>) egy vízszintes menü, amely különböző aloldalakra mutató linkeket tartalmaz. A chartjs.html fájl az aktív oldal, amit vizuálisan is kiemel a navigációban.

A fő tartalmi rész (<div class="container">) egy öt sorból és hat oszlopból álló táblázatot tartalmaz, ahol az első oszlop a sorokat azonosítja, a többi pedig számértékeket. Ez a táblázat manuálisan, statikus HTML-ben van definiálva. A #dataTable azonosítóval ellátott <table> elem szolgál a vizualizált adatok forrásául.

A JavaScript az oldal alján található, közvetlenül a </body> tag előtt. A kód első része eseményfigyelőt állít be a táblázat egészére. Amikor a felhasználó egy táblázatsorra kattint, a JavaScript az adott sor celláit lekéri, és kiszűri az első cellát (amely csak a sor sorszáma). A többi cella értékét egész számként egy tömbbe gyűjti.

Ez a tömb a updateChart() függvényhez kerül, amely a Chart.js segítségével generál vonaldiagramot. A korábbi diagram (ha már létezett) először törlésre kerül, hogy ne maradjon duplikált vagy régi ábra a vásznon. A diagram típusa line, vagyis vonaldiagram, a vízszintes tengely címkéi statikusak és az oszlopneveknek felelnek meg. A datasets mező tartalmazza az aktuálisan kiválasztott sorból származó adatokat. A diagram stílusa enyhén lekerekített (tension: 0.1) és türkiz színnel jelenik meg.

A JavaScript defer attribútummal van betöltve, így a chartjs.js fájl csak azután töltődik be, hogy az oldal teljes HTML-tartalma már betöltődött, ezzel elkerülve az esetleges szkript-hibákat. A Chart.js könyvtár egy CDN-en keresztül érhető el, így nincs szükség külön letöltésre vagy telepítésre.

Az oldal stílusát CSS szabályok határozzák meg, amelyek alapvetően reszponzív elrendezést biztosítanak. A fő tartalom 80%-os szélességű konténerben helyezkedik el, középre igazítva. A navigációs sáv sötétszürke háttérrel és zöld kiemeléssel reagál az aktív vagy föléhúzott elemekre. A táblázat dizájnja letisztult, vékony szegélyekkel és halvány háttérrel. a drawChart(row), amely a kiválasztott HTML táblázatsor alapján rajzolna vonaldiagramot. Ez a függvény független a fenti működéstől, és nem hivatkozik rá semmi a HTML kódban, így valószínűleg egy alternatív vagy korábbi verzió része volt. Funkcionálisan hasonló a updateChart() függvényhez, de nem kapcsolódik aktívan a többi logikához, így akár törölhető is.

A képen szöveg, képernyőkép, diagram, sor látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, Diagram, sor látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, diagram, szoftver látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

Ajax.html/ajax.js/ajaxapi.php

A rendszer célja, hogy demonstrálja, hogyan lehet JavaScript és PHP segítségével külső API-val kommunikálni a CRUD (Create, Read, Update, Delete) műveletek megvalósításához, miközben a felhasználó számára valós idejű visszajelzést biztosít. A HTML oldal HTML5 elemekkel és CSS stíluslapokkal lett kialakítva, figyelembe véve a reszponzív és könnyen áttekinthető elrendezést.

A képen szöveg, képernyőkép, Betűtípus, képernyő látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, sor látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A weboldal fejléce egyértelműen jelzi, hogy egy egyetemi házi feladatról van szó, amely a Web-programozás-1 előadáshoz kapcsolódik. A navigációs sáv különféle témakörök (mint például ChartJS, React, HTML5 példák stb.) közötti váltást tesz lehetővé, miközben vizuálisan is visszajelzést ad arról, hogy épp melyik oldalon tartózkodik a felhasználó az "active" osztály segítségével. Az oldal háttere világos, a fő tartalmak világos dobozokban, szegéllyel elválasztva jelennek meg, ezáltal jól elkülönülnek egymástól és könnyen értelmezhetőek.

A központi funkció az **AJAX alapú API-kezelés**, amely egy ajax.js nevű JavaScript fájlon keresztül történik. Ez a fájl tartalmazza az összes fontos funkciót: az adatok beolvasását (read), létrehozását (create), módosítását (update), valamint törlését (delete). A JavaScript aszinkron fetch API-t használ a HTTP POST kérések lebonyolítására, így a felhasználónak nem kell újratöltenie az oldalt minden adatfrissítéshez. Ez a felhasználói élmény szempontjából kulcsfontosságú, és egyre elterjedtebb a modern webes alkalmazásoknál.

Az ajax.js fájl elején egy hardcoded azonosító (code = "AAAAAAabc123") található, amelyet minden kéréshez hozzáfűznek. Ez valószínűleg azonosításként vagy jogosultság-ellenőrzésként szolgál az API felé. Az url változó a ajaxapi.php fájlra mutat, amely a háttérben hívja meg a valódi API-t egy cURL kéréssel. Ez a PHP proxy megoldás biztonságosabbá teszi a kommunikációt, mivel a kliensoldal nem kommunikál közvetlenül az API-val, és így az API címe vagy logikája elrejthető.

A read() függvény lekéri a rekordokat az API-tól, és azokból egy táblázatot épít fel HTML-ben. Ezen kívül a magasság mezőből statisztikát is készít: összegzi az értékeket, kiszámolja az átlagot, és meghatározza a legnagyobb értéket. Ez nemcsak informatív, de azt is mutatja, hogyan lehet egyszerű matematikai műveleteket végezni JavaScript segítségével. A create(), update() és deleteF() függvények a klasszikus CRUD műveleteket valósítják meg, ügyelve az űrlapmezők validációjára is – például, hogy a mezők nem lehetnek üresek, és nem lehetnek hosszabbak 30 karakternél. Sikeres vagy sikertelen műveletek után a rendszer visszajelzést ad a felhasználónak, illetve frissíti a táblázatot, hogy az aktuális adatok legyenek láthatók.

Az update() előtt található getDataForId() függvény lehetővé teszi, hogy a megadott ID alapján lekérjük az adott rekord adatait, és automatikusan kitölti az űrlapmezőket. Ez intuitívvá teszi a szerkesztést, mivel a felhasználónak nem kell kézzel újra beírnia az értékeket.

A háttérben futó ajaxapi.php egy PHP szkript, amely POST kéréseket fogad. Az összes szükséges mezőt összegyűjti egy asszociatív tömbbe, majd cURL segítségével elküldi ezeket a http://gamf.nhely.hu/ajax2/ címre. Ez a cím feltehetően egy, az egyetem által biztosított oktatási API, amely a kérések alapján különféle válaszokat küld vissza JSON formátumban. A PHP szkript ezeket a válaszokat egyszerűen visszaküldi a JavaScript-nek, amely feldolgozza és megjeleníti őket.

Az oldal vizuális struktúrája három fő "actionDiv" szekcióra van bontva, mindegyik egy CRUD művelethez kapcsolódik. Ezek egymás mellett helyezkednek el, szimmetrikusan, jól elkülönítve, ami segíti a tájékozódást. Az elrendezés egyszerű, mégis átlátható, és lehetőséget ad a különféle műveletek párhuzamos kezelésére.

Az oldal alján található egy lábléc, amely az oldal készítőjére utal, névvel és Neptun-kóddal. Ez az egyetemi beadandók esetében megszokott gyakorlat, és egyértelműen dokumentálja a projekt tulajdonosát.

Oojs.html/oojs.js

Ez a weboldal egy objektumorientált JavaScript (OOJS) alkalmazásának. A HTML és CSS része a struktúrát és megjelenést biztosítja, míg a JavaScript a dinamikus működést és az objektumorientált szemléletet demonstrálja. Az oldal célja egy tanulási projekt bemutatása, amelyben az objektumorientált programozás koncepcióit a webes környezetbe integrálják, miközben felhasználóbarát és vizuálisan is átgondolt módon tálalják a tartalmat.

A weboldal felépítése egy klasszikus elrendezést követ: van egy fejléc, amely tartalmazza a főcímet, egy navigációs sáv, amely lehetővé teszi az oldal különböző részeinek elérését, egy fő tartalomterület, ahol az aktuális modul jelenik meg, valamint egy rögzített lábléc, amelyben az oldal készítőjének neve és Neptun kódja szerepel. A navigációs sáv sötét színvilágú, és a kiválasztott menüpont zöld háttérszínt kap, ezzel is segítve a felhasználói tájékozódást.

A CSS modern, reszponzív szemléletű. Flexboxot használ a navigáció és a gombok elrendezésére, valamint box-shadow és border-radius stílusokkal vizuálisan is kiemeli a menüpontokat és az interaktív elemeket. A gombok stílusa reszponzív, lebegéskor árnyalattal változnak, és lekerekített sarkokkal kellemes vizuális visszajelzést nyújtanak.

A JavaScript a Menu osztály köré épül, amely egy pontgyűjtő játékot valósít meg. A menüben különböző gombok szerepelnek, amelyek kattintásra pontokat adnak a felhasználónak, a pontszámot pedig a felület folyamatosan frissíti. A Menu osztály képes létrehozni saját HTML konténerét, gombokat ad hozzá, és a pontszámot is frissíti. A SpecialMenu osztály örökli a Menu osztályt, és kiegészül egy bónusz értékkel, bár ez jelenleg a példában még nem kerül közvetlen felhasználásra, de bemutatja az öröklődés működését. A menü rugalmasan bővíthető új gombokkal, mivel az addButton metódus egyszerűen kezelhető és dinamikusan hozza létre a felület elemeit.

A képen szöveg, képernyőkép, szoftver, Weblap látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, Operációs rendszer látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, tervezés látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A JavaScript második szegmense egy külön demonstrációja az objektumorientált JavaScript másik lehetséges alkalmazásának: a könyvtárkezelés modelljének. Három osztály szerepel itt: az Item, a Book és a Library. Az Item egy általános szülőosztály, amely egy nevet tárol, és megjelenítő metódussal rendelkezik. A Book osztály ezt bővíti azzal, hogy egy szerzőt is hozzárendel, és ezt a megjelenítés során kiírja. A Library osztály egy kollekciót kezel, könyveket tárol, és képes azokat egyenként megjeleníteni. A displayBooks metódus minden könyvet végigiterál és megjelenít a felhasználó számára.

Ez a második rész a klasszikus OOP szemléletet emeli ki, a JavaScriptben használt class, constructor, extends, super kulcsszavakon keresztül, és megmutatja, hogyan lehet DOM-manipulációval együtt dolgozni objektumorientált módon. Minden könyv DOM-elemként jelenik meg, egy div elembe ágyazva, ami lehetővé teszi a vizuális reprezentációt is.

React.html/react.js

Ez a weboldal egy React alkalmazást mutat be, amelynek célja az alapvető React-fogalmak demonstrálása. A teljes oldal HTML-ben van felépítve, modern CSS dizájnnal rendelkezik, amely következetesen illeszkedik a többi oldal struktúrájához, és egy külön JavaScript szkripttel – react\_app.js néven – kezeli a React-funkcionalitást. A rendszer bemutatja, hogyan lehet egy React alkalmazást integrálni egy statikus HTML oldalba anélkül, hogy bonyolult build-eszközöket vagy fejlesztőkörnyezeteket használnánk, így kifejezetten alkalmas tanulásra és alapvető prototípus-készítésre.

A képen szöveg, képernyőkép, Betűtípus, szoftver látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

A HTML rész logikusan strukturált: a header tartalmazza az oldal címét, a nav egy menüt, amely segíti a navigációt a projekt különböző aloldalai között. A „React” menüpont aktívként jelenik meg, így a felhasználó vizuálisan is látja, hogy épp ebben a részben van. A #content div tartalmazza a címmezőt és a #root elemet, amely a React alkalmazás gyökércsomópontjaként szolgál – ide kerül a dinamikusan generált React-tartalom. A footer egységesen szerepel az oldalon, biztosítva, hogy minden oldal tartalmazza a készítő nevét és a Neptun-kódot.

A React alkalmazás a Counter komponens köré épül. Ez egy funkcionális komponens, amely a useState hookot használja, hogy lokális állapotot kezeljen – jelen esetben a számláló értékét. A count változó tartalmazza a jelenlegi értéket, a setCount pedig annak frissítésére szolgál. Két gomb szerepel a komponensben: az egyik növeli, a másik csökkenti az értéket. A gombokra történő kattintás hatására a komponens újrarenderelődik, a számláló pedig frissül. Ez egy nagyon jó példa arra, hogyan működik a komponensalapú gondolkodás Reactben, és hogyan lehet állapotot (state) kezelni egyszerűen és hatékonyan.

Az App komponens egyetlen szülőkomponens, amely a Counter komponenst tartalmazza. Ezáltal a kód jól strukturált, és lehetőséget ad a további komponensek bővítésére a jövőben. A komponensek JSX szintaxist használnak, ami a JavaScript és HTML kombinációját teszi lehetővé, olvasható és intuitív módon. A JSX futtatásához a Babel könyvtárat használják, amely a modern JavaScript szintaxist futtathatóvá teszi a böngésző számára.

A React és ReactDOM könyvtárak CDN-en keresztül kerülnek betöltésre, ami lehetővé teszi, hogy az egész alkalmazás build rendszer nélkül fusson.