JEGYZŐKÖNYV

Web technológia alapjai

GameDev weboldal

Készítette: **Danyi Károly**

Neptunkód: **RDC1L4**

Dátum: 2024.május.05

Miskolc, 2024.

Tartalomjegyzék

Miskolc, 2024.

Bevezetés

Ez a projekt egy olyan online platformot céloz meg, amely a játékfejlesztés világába nyújt betekintést és segítséget nyújt mindazoknak, akik érdeklődnek a játékfejlesztés iránt. A weboldal célja, hogy összegyűjtse és bemutassa a legfontosabb információkat a különböző gametípusokról és game engine-ekről, valamint segítséget nyújtson azoknak, akik szeretnék elkezdeni a saját játékok fejlesztését. A gametípusok és game engine-ek áttekintése fontos része ennek a dokumentációnak. A gametípusok olyan kategóriák, amelyekbe a játékok csoportosíthatók, és különböző jellemzők alapján különíthetők el egymástól. Ezek a kategóriák segítenek megérteni a különféle játékstílusokat és műfajokat, amelyekre a fejlesztők ötleteiket építhetik. A game engine-ek pedig olyan szoftveres eszközök, amelyek lehetővé teszik a fejlesztők számára, hogy játékokat hozzanak létre, megkönnyítve ezzel a grafikai, fizikai és egyéb technikai kihívások kezelését. A dokumentáció további részeiben részletesen bemutatjuk a különböző gametípusokat és game engine-eket, valamint útmutatást nyújtunk azoknak, akik saját játékfejlesztési projektbe szeretnének kezdeni. Reméljük, hogy ez a weboldal inspirációt és segítséget nyújt mindazoknak, akik érdeklődnek a játékfejlesztés iránt, és lehetőséget biztosít arra, hogy saját ötleteiket valóra váltsák a digitális játékok világában

Miskolc, 2024.

1.Fájl Lista

1.1.HTML fájlok:

- index.html
- form.html
- engine.html
- games.html
- doc.html

1.2.CSS fájlok:

- gamesstyles.css
- styles.css
- formstyless.css
- stylesJQ.css

1.3.JS fájlok

- RDC1L4_1.js
- server.js

1.4.JSON fájlok

- package.json
- Json.json

1.5.Egyéb fájlok

- RDC1L4.pdf
- Gamemakerlogo.png
- godot-logo.png
- Unity-logo.jpg
- Unreal_Engine.png
- StardewV.jpg
- Terraria.jpg
- StardewValleyTrailer.mp4
- TerrariaTrailer.mp4

Miskolc, 2024.

2.Index.html

Weboldal Szerkezete:

A GameDev weboldal HTML, CSS és JavaScript nyelveken épül fel.

Fejléc (Header):

- A fejlécben található a navigációs sáv, amely lehetővé teszi a felhasználók számára, hogy könnyen elnavigáljanak az oldal különböző részei között.
- A navigációs sáv tartalmazza a következő linkeket: Főoldal, Játékok, Fejlesztő környezetek, Tipp kérő és Dokumentum.

Fő Tartalom (Main):

- A fő tartalom rész három fő szakaszból áll: "Bemutató", "Pixel Art Játék" és "2D Játék". A " Bemutató " szakasz bemutatja a GameDev weboldalt és annak főbb jellemzőit.
- A "Pixel Art Játék" és "2D Játék" szakaszok részletesen ismertetik ezeket a játékfajtákat, jellemzőiket és példákat mutatnak be.

Videó Bemutató (Video):

- A videó bemutató szakaszban található egy beágyazott videólejátszó, amely lehetővé teszi a felhasználók számára, hogy megnézzék a kiválasztott játékok trailerjeit.
- A videólejátszó alatt gombok találhatók, amelyek segítségével a felhasználók váltogathatnak a rendelkezésre álló videók között, lejátszhatják, szüneteltethetik vagy újraindíthatják azokat.

Lábjegyzet (Footer):

- A lábjegyzet tartalmazza a weboldal szerzőjének nevét és egyéb dolgokat.

Technikai Megvalósítás:

- A weboldal HTML5, CSS3 és JavaScript segítségével készült.
- A Bootstrap keretrendszert használja a responszív elrendezés és a stílusok kezeléséhez.
- A videólejátszóhoz szükséges funkciókat JavaScript segítségével valósították meg.

```
<script>
        var videoElement = document.getElementById("gameVideo");
        var videos = ["StardewValleyTrailer.mp4",
"TerrariaTrailer.mp4"];
        function playPause() {
            if (videoElement.paused) {
                videoElement.play();
            } else {
                videoElement.pause();
            }
        }
        function restartVideo() {
            videoElement.currentTime = 0;
            videoElement.play();
        }
        function playVideo(source) {
            videoElement.src = source;
            videoElement.play();
        }
        var currentVideoIndex = 0;
        videoElement.addEventListener('ended', function() {
```

```
            currentVideoIndex = (currentVideoIndex + 1) %
videos.length;
            videoElement.src = videos[currentVideoIndex];
            videoElement.play();
        });
    </script>
```

playPause(): Ez a függvény ellenőrzi, hogy a videó lejátszása szüneteltetett-e vagy sem. Ha a videó szüneteltetett állapotban van, akkor elindítja, ha pedig éppen lejátszás alatt van, akkor szünetelteti.

restartVideo(): Ez a függvény a videó lejátszását nullára állítja, majd elindítja újra a videót. Tehát visszatekeri a videót az elejére és újra lejátsza.

playVideo(source): Ez a függvény egy adott videót játszik le. A source paraméter egy videó forrását várja, és amikor meghívják ezt a függvényt, a videoElement az adott forrású videót fogja lejátszani. ended eseménykezelő: Ez az eseménykezelő akkor fut le, amikor a videó végére ér. Ekkor a kód az aktuális videó indexét növeli egyel modulo a videók számával, így körbejárja a videók tömbjét. Majd beállítja a videoElement forrását az újra kiválasztott videóra, majd elindítja azt.

Tehát a kód egy egyszerű videólejátszót valósít meg két videóval (StardewValleyTrailer.mp4 és TerrariaTrailer.mp4), és lehetővé teszi ezek lejátszását, megállítását, újrakezdését, valamint automatikusan továbblépteti a következő videóra, amikor az aktuális videó véget ér.

3.games.html

3.1.Weboldal Szerkezete:

A Games weboldal HTML és CSS nyelveken épül fel.

Fejléc (Header):

- A fejlécben található a GameDev logó és a navigációs sáv, amely lehetővé teszi a felhasználók számára, hogy könnyen elnavigáljanak az oldal különböző részei között.
- A navigációs sáv tartalmazza a következő linkeket: Főoldal, Játékok, Fejlesztő környezetek, Tipp kérő és Dokumentum.

Fő Tartalom (Main):

- A fő tartalom rész kettő fő szakaszból áll: "Játék típusok bemutató" és "Táblázat" részből.

Bemutató:

- Különböző játéktípusokat mutat be, nagyon látványos CSS formázással és interaktív elemekkel.

Táblázat:

- Egy játékfejlesztési táblázat, amely különböző játéktípusokat sorol fel, és ezeknek a típusoknak a fejleszthetőségét, népszerűségét, grafikai komplexitását és játékmenet bonyolultságát mutatja be. Többnyire irányadónak van tervezve.

CSS:

- A CSS tartalmazza az alap beállításait az oldalnak.
- Tartalmazza a kártyák teljeskörű animációit, az adott oldalon.
- A html kódban és a css fájlban is van CSS formázás.
- Tábla stílusának formázását is tartalmazza

Kódrészlet:

Kártya animációk részlet:

```
.wrap {
display: flex;
flex-wrap: nowrap;
justify-content: space-between;
width: 85vmin;
height: 65vmin;
margin: 2rem auto;
border: 8px solid;
border-image: linear-gradient(
-50deg,
green,
#00b300,
forestgreen,
green,
lightgreen,
#00e600,
green
)
1;
transition: 0.3s ease-in-out;
position: relative;
overflow: hidden;
}
.overlay {
position: relative;
display: flex;
width: 100%;
height: 100%;
padding: 1rem 0.75rem;
```

```
background: #186218;
transition: 0.4s ease-in-out;
z-index: 1;
}
.overlay-content {
display: flex;
flex-direction: column;
justify-content: space-between;
width: 15vmin;
height: 100%;
padding: 0.5rem 0 0 0.5rem;
border: 3px solid;
border-image: linear-gradient(
to bottom,
#aea724 5%,
forestgreen 35% 65%,
#aea724 95%
)
0 0 0 100%;
transition: 0.3s ease-in-out 0.2s;
z-index: 1;
}
.image-content {
position: absolute;
top: 0;
right: 0;
width: 70vmin;
height: 100%;
background-image: url("Call-of-Duty.jpg");
background-size: cover;
transition: 0.3s ease-in-out;
/* border: 1px solid green; */}
```

A .wrap osztály egy flexbox konténert definiál, ami nem engedi a gyerekelemeit sortörésre, középre igazítja őket, egy adott méretű keretet (85vmin szélesség, 65vmin magasság), valamint átmenetet alkalmaz a változásokra és az overflow tulajdonsággal lehetőséget ad a tartalom elrejtésére, ha az túlmutat a kereten.

Az .overlay osztály egy másik flex konténert hoz létre, amelynek gyerekelemei között az elrendezés oszlop alapú, az áttűnése 0.4 másodpercig tart.

Az .overlay-content osztály egy további flex konténert definiál, amely a tartalom megjelenítésére szolgál az overlay-en belül. A tartalom megjelenése animált, és lineáris színátmeneteket használ a keret különböző részein.

Az .image-content osztály egy abszolút pozicionált elem, ami a háttérképet tartalmazza a .wrap elemen belül. Az áttűnése 0.3 másodpercig tart.

Összességében ez a CSS kód egy dinamikus, áttűnő elrendezést hoz létre, amelynek részei egymásra épülnek és animáltak, így egy látványos felhasználói élményt biztosítanak.
A fenti kód, csak egy részlet, de már ebből is látható hogy nagyon komplex az animációknak a beállítása és a kártyák stílusának szerkesztése.
Ez mind a jó megjelenés érdekében került bele.

Miskolc, 2024.

3.2.Táblázat formázás:

```
table {
    width: 100%;
    border-collapse: collapse;
    border: 2px solid #336633;
    margin-bottom: 20px;
}

th, td {
    padding: 12px;
    text-align: left;
    border-bottom: 1px solid #336633;
}

th {
    background-color: #336633;
    color: white;
    font-weight: bold;
    text-transform: uppercase;
}
```

A table, szabályok beállítják a táblázat szélességét 100%-ra, összeomló határvonalat alkalmaznak a cellák között, 2px vastag zöld szegélyt adnak a táblázat köré, és 20px margót állítanak be az alatta következő elemhez.

A th, td szabályok meghatározzák a fejléc- (th) és adatcellák (td) formázását.

Mindkettőre 12px-es belső margót állítanak be, a szöveget balra igazítják, és 1px vastag zöld szegélyt adnak az aljukhoz.

A th szabályok beállítják a fejlécek háttérszínét zöldre, a szöveg színét fehérre, félkövérré teszik a betűket, és nagybetűsre alakítják a szöveget. Ezáltal a fejléc

cellák kiemelkednek és könnyen azonosíthatók a többi cellától.Tartalmazza az alapvető beállításokat. Egyszerű formázás, a letisztultság érdekében.

TÍPUS	FEJLESZTHETŐSÉG (S-F)	NÉPSZERŰSÉG (1-5)	GRAFIKA KOMPLEXITÁSA (1-5)	JÁTÉKMENET BONYOLULTSÁGA (1-5)
Akció	B	5	4	3
Souls-like	A	4	5	4
Kaland	B	5	5	3
Szerepjáték	C	5	4	4
Stratégiai	C	3	3	5
Competitive	A	4	3	2
Szimuláció	C	3	4	3
Horror	D	4	4	3
Platformer	B	4	3	4
Sport	B	3	3	3
Indie	C	3	2	3
MMO	F	5	5	5

Ez a CSS kód felel a táblázat megjelenéséért. Beállítja a táblázat szélességét 100%-ra, hogy teljesen kitöltse a rendelkezésre álló teret. A táblázat körvonalát egy vékony, zöld színű vonal határozza meg. A cellák belső térközét, az igazítást és a vonalakat állítja be, hogy tisztábbá tegye a táblázatot. A táblázat fejlécének háttérszíne zöld lesz, hogy kiemelje az elnevezéseket, és a szöveg fehér színű lesz, hogy jól látható legyen a zöld háttéren. Egyszerű, de hatékony stílusokat alkalmaz a táblázat megjelenítésére.

Miskolc, 2024.

4.Form.html

4.1.Html kód

Ez egy HTML kód, amely egy űrlapot hoz létre, amelyet "Tipp kérő"-nek neveznek. Az űrlapon a felhasználók játékokkal kapcsolatos információkat tudnak megosztani.

Az űrlapnak több mezője van, mint például:

- játékcím,
- játék leírása,
- játéktípus,
- platformok,
- fejlesztő környezet,
- játék színe ,
- kiadás dátuma.

Ezeket a mezőket megfelelően formázza és validálja a JavaScript kód. A weboldalnak van egy navigációs sávja is, amely lehetővé teszi a felhasználók számára, hogy könnyen navigáljanak más oldalakra a webhelyen.

Ezenkívül van egy AJAX hívás is, amely egy JSON fájlt tölt be és megjeleníti a weboldalon található játékok adatait.

Összességében ez az HTML kód egy felhasználóbarát űrlapot és navigációs sávot kínál a felhasználóknak, hogy megoszthassák és böngészhessék a játékokkal kapcsolatos információkat.

4.2.Form JS

A JavaScript kód egy validateForm nevű függvényt definiál, amelyet az űrlap validálására használnak. A függvény ellenőrzi az űrlap különböző mezőinek tartalmát, például a játékcímet, a leírást, a típust, a kiválasztott platformokat és fejlesztő környezeteket, valamint a kiadás dátumát. Ha bármelyik mező nem felel meg a validációs követelményeknek (például üres vagy nem megfelelő formátumú), akkor a függvény beállítja a megfelelő hibaüzeneteket és visszatérési értékként false-t ad vissza, jelezve, hogy az űrlap nem lett sikeresen validálva.

Ha minden mező megfelel a követelményeknek, akkor a függvény visszatérési értéke true, és egy sikeres küldés üzenetet jelenít meg.

A kód továbbá tartalmaz még néhány segédmetódust, például a changeColor, amely az űrlap egyes elemeinek háttérszínét változtatja a felhasználó által kiválasztott érték alapján, valamint a resetColors, amely visszaállítja az űrlap elemeinek stílusát és törli a hibaüzeneteket.

Végül a kód tartalmaz egy AJAX hívást is, amely egy JSON fájlt tölt be a játékok adataival, és ezeket megjeleníti a weboldalon. Ez lehetővé teszi a felhasználók számára, hogy böngésszenek és megtekintsék az eddigi bevitt adatokat.

```
$.ajax({
            url: 'json.json', // A JSON fájl elérési útja
            dataType: 'json',
            success: function (data) {
                // Az adatok megjelenítése
                var jatekokDiv = $('#jatekok');
                $.each(data.jatekok, function (index, jatek) {
                    var jatekHtml = `
                    <div>
                        <h2>${jatek.cim}</h2>
                        <p><strong>Leírás:</strong>
${jatek.leiras}</p>
                        <p><strong>Típus:</strong>
${jatek.tipus}</p>
                        <p><strong>Platformok:</strong>
${jatek.platformok.join(', ')}</p>
                        <p><strong>Környezet:</strong>
${jatek.kornyezet}</p>
                        <p><strong>Szín:</strong> <span
style="background-color:
${jatek.szin};">${jatek.szin}</span></p>
                        <p><strong>Kiadás dátuma:</strong>
${jatek.kiadas_datuma}</p>
                    </div>
                `;
                    jatekokDiv.append(jatekHtml);
                });
            },
            error: function (xhr, status, error) {
                console.error('Hiba történt a JSON fájl
betöltése közben:', error);
            }
        });
```

Ez a kód egy AJAX kérést indít egy JSON fájlhoz, amely adatokat tartalmaz játékokkal kapcsolatban. Amikor a kérés sikerrel teljesül (success esetén), a visszakapott adatokat megjeleníti a weboldalon.

Az AJAX kérés a json.json elérési útvonalon keresztül éri el a JSON fájlt.

A dataType: 'json' beállítás meghatározza, hogy a visszakapott adatok JSON formátumban várhatók.

A success eseménykezelő függvényben a visszakapott adatok feldolgozása és megjelenítése történik. A data változó tartalmazza a visszakapott JSON objektumot.

A .each() függvény segítségével végigmegyünk minden játékon a data.jatekok tömbben, majd azok adatait megjelenítjük az oldalon.

A játékok adatait HTML elemekbe szervezzük, például <h2> címkékkel, <p> bekezdésekkel, és a játék adatait behelyettesítjük az objektumból származó adatokkal, például a cím, leírás, típus stb.

A megjelenített adatokat a jatekokDiv változóba gyűjtött HTML elemhez fűzzük hozzá a .append() metódussal.

Ha a kérés során valamilyen hiba történik (error esemény), a hibát a konzolra írja ki a console.error() segítségével.
A kód JavaScriptben íródott, és a jQuery nevű JavaScript könyvtárat használja egy AJAX kérés végrehajtásához. Az AJAX kérés JSON adatokat kér le egy meghatározott URL-ről. A kódban magyar nyelvű kommentek találhatók, amelyek leírják a különböző részeket, például az AJAX hívás inicializálását és a siker és hiba kezelését. A siker esetén HTML elemek dinamikusan létrejönnek és adatokkal lesznek feltöltve a JSON válaszból.

```
function validateForm() {
            var regex = /^[A-Za-z0-9\s]+$/;
            var title =
document.getElementById("gameTitle").value;
            var description =
document.getElementById("gameDescription").value;
            var type =
document.getElementById("gameType").value;
            var platforms =
document.querySelectorAll('input[name="platform"]:checked');
            var asd =
document.querySelectorAll('input[name="radiobutton"]:checked');
            var color =
document.getElementById("gameColor").value;
            var releaseDate =
document.getElementById("releaseDate").value;
            var isValid = true;


            if (title == "" || !regex.test(title)) {
        document.getElementById("gameTitle").style.border = "2px
solid red";

document.getElementById("gameTitleError").style.color = "red";
                var labelElement =
document.getElementById("gameTitleError");
                labelElement.innerHTML = "Hibás Adat";
                isValid = false;
            } else {
         document.getElementById("gameTitleError").style.color
="green";
       document.getElementById("gameTitle").style.border = "2px
solid green";
                var labelElement =
document.getElementById("gameTitleError");
                labelElement.innerHTML =
"<strong>Játékcím:</strong>";
            }
```

Miskolc, 2024.

Ez a JavaScript függvény ellenőrzi egy űrlap mezőinek validitását, amelyek játék adatokat képviselnek.

- A függvényben különböző változókban tárolja az űrlap különböző mezőinek értékeit, például a játék címét, leírását, típusát, platformjait, stb.
- A regex változóban egy reguláris kifejezés van tárolva, amely az elfogadható karaktereket definiálja a cím mezőben (csak betűk, számok és szóközök).
- A isValid változó segítségével nyomon követi, hogy az ellenőrzés során volt-e valamilyen hiba.
- Az if-else szerkezetek segítségével ellenőrzi a mezők validitását:
    - Ha a cím mező üres vagy nem felel meg a megadott reguláris kifejezésnek, akkor piros színű keretet ad hozzá, kiír egy hibaüzenetet és beállítja az isValid változót hamisra.
    - Ha a cím mező helyes, akkor zöld színű keretet ad hozzá, és megjeleníti a cím mezőhöz tartozó címkét zöld színben.
- A többi mező validációját nem látjuk a kódrészletben, de valószínűleg hasonló módon történik azok ellenőrzése is.

- A függvény különböző bemeneti mezőket, például a ‘gameTitle’, ‘releaseDate’ és ‘platforms’ mezőket ellenőrzi az értékeikre, és feltételes formázást alkalmaz annak alapján, hogy a bemenetek megfelelnek-e bizonyos kritériumoknak. Például, ha a ‘gameTitle’ mező üres, akkor a hozzá tartozó címkét pirosra változtatja és hibaüzenetet jelenít meg.

5.Engine.html

5.1.Szerkezet

A weboldal fő részében négy fejlesztői környezetet mutat be:

- Unity,
- Unreal Engine,
- Godot Engine,
- GameMaker Studio.

Mindegyik fejlesztői környezethez tartozik egy gomb, amelyek segítségével további információkhoz juthatunk.

Minden fejlesztői környezetet egy kép és egy rövid leírás kísér, ami segít megérteni, hogy miért népszerű és milyen lehetőségeket kínál a játékfejlesztőknek.

A leírások az adott fejlesztői környezet erősségeire és felhasználóbarát jellegére fókuszálnak, és könnyen érthető módon fogalmazzák meg az információkat.

A weboldal stílusa egyszerű és letisztult, a gombok és a szövegek könnyen észrevehetők és érthetők.

A képek és a színek segítségével vonzóvá teszik az oldalt és segítenek az információk megjelenítésében. A navigációs sáv és a gombok áttekinthetővé teszik az oldalt, így a felhasználók könnyen eligazodhatnak és megtalálhatják az érdeklődésüknek megfelelő információkat.

5.2.JQuery és js Animáció

Ezen a weboldalon található szinte összes animáció, JQuery és js-el van megoldva.

Amikor az oldalon a "Unity", "Unreal Engine", "Godot Engine" vagy "GameMaker Studio" gombra kattintunk, akkor azokhoz tartozó részek megjelenése megváltozik animációval.

Például a gombra kattintva az adott rész elmozdul és nő, majd újra kattintva visszatér az eredeti méretéhez és helyzetéhez.

Emellett egy másik gombra kattintva az oldalsáv megnyitható vagy bezárható, ami további interaktivitást tesz lehetővé az oldalon.

Ezen kívül négy másik gomb lehetővé teszi az információk megjelenítését vagy elrejtését a különböző fejlesztői környezetekhez tartozó leírásokban.

Ez a kód segít a felhasználónak az oldalon való navigálásban és az információk könnyű megjelenítésében és elrejtésében az adott témákhoz kapcsolódóan.

5.3.Kód magyarázat animációhoz

```
    $("#unity2").click(function () {
        if (!isrOpen) {
            $("#unidiv2").animate({ left:'700px', width:
'400px', height:'500px', fontSize: '12pt' }, 1000);
            isrOpen = true;
        } else {
            $("#unidiv2").animate({ left: '100%', width: '-
250px', height:'0px', fontSize: '12pt' }, 1000);
            isrOpen = false;
        }
    });
```

Ez a JavaScript kód egy jQuery eseménykezelőt definiál a #unity2 elemre. Amikor a #unity2 elemre kattintanak, az eseménykezelő vizsgálja, hogy az isrOpen változó értéke igaz-e vagy sem.

- Ha az isrOpen értéke hamis, azaz az elem nincs nyitva, akkor az #unidiv2 elemet animáltan elmozdítja 700px-re balra, megváltoztatja a szélességét 400px-re, a magasságát pedig 500px-re, és a betűméretét 12pt-re 1000 milliszekundum alatt. Az isrOpen változót igazzá állítja, jelezve, hogy az elem mostantól nyitva van.
- Ha az isrOpen értéke igaz, azaz az elem nyitva van, akkor az #unidiv2 elemet animáltan elmozdítja 100%-ra balra (azaz az ablakon kívülre), megváltoztatja a szélességét -250px-re (tehát az elem szélessége csökken), a magasságát 0px-re (tehát az elem magassága nulla lesz), és a betűméretét 12pt-re 1000 milliszekundum alatt. Az isrOpen változót hamissá állítja, jelezve, hogy az elem mostantól zárva van.

Ez a kód tehát egy animált fel- és lecsúszó hatást hoz létre az #unidiv2 elemmel, amikor a #unity2 elemre kattintanak, és állítja az isrOpen változó értékét annak

megfelelően, hogy az elem nyitva vagy zárva van. Gombra kattintáskor megnézi a jelenlegi helyét a div-nek, egy boolean típusú változóval. Az adott helytől függ hogy melyik animáció megy végbe.

```
    $("#oldalsav").click(function () {
        if (!OldalisOpen) {
            $('.sidePanel').toggleClass('show');
            $('.sidePanel2').toggleClass('show');
            OldalisOpen = true;
        } else {
            $('.sidePanel').removeClass('show');
            $('.sidePanel2').removeClass('show');
            OldalisOpen = false;
        }
    });
```

Ez a JavaScript kód egy jQuery eseménykezelőt definiál az #oldalsav elemre. Amikor az #oldalsav elemre kattintanak, az eseménykezelő vizsgálja, hogy az OldalisOpen változó értéke igaz-e vagy sem.

- Ha az OldalisOpen értéke hamis, azaz az oldalsávak nincsenek nyitva, akkor az .sidePanel és .sidePanel2 osztályú elemekre alkalmazza a show osztályt azok megjelenítéséhez. Az OldalisOpen változót igazzá állítja, jelezve, hogy az oldalsávak mostantól nyitva vannak.
- Ha az OldalisOpen értéke igaz, azaz az oldalsávak nyitva vannak, akkor az .sidePanel és .sidePanel2 osztályú elemekről eltávolítja a show osztályt azok elrejtéséhez. Az OldalisOpen változót hamissá állítja, jelezve, hogy az oldalsávak mostantól zárva vannak. Ez a kód tehát egy oldalsávak megjelenítését és elrejtését vezérli az OldalisOpen változó segítségével, amikor az #oldalsav elemre kattintanak. Az oldalsávak megjelenését a show osztállyal valósítja meg, amelyet a .toggleClass() vagy a .removeClass() metódusokkal ad hozzá vagy távolít el az elemekről. Az

oldalsávokat lehet kinyitni vele és bezárni. Ezt mind egy gomb kattintással. Itt is az előző technikát is alkalmazzuk.

```
$("#ub").click(function(){
  $("#ut").toggle(1000);
});

$("#unb").click(function(){
    $("#unt").toggle(1000);
  });

  $("#godb").click(function(){
    $("#godt").toggle(1000);
  });

$("#gmb").click(function(){
    $("#gmt").toggle(1000);
});
```

Ez a JavaScript kód négy különböző jQuery eseménykezelőt definiál. Minden eseménykezelő egy gombra (#ub, #unb, #godb, #gmb) kattintva egy-egy másik elemet (#ut, #unt, #godt, #gmt) jelenít meg vagy rejteget el.

- Az #ub gombra kattintva az #ut elem megjelenítése vagy elrejtése történik, animációval 1000 milliszekundumos időtartamban.
- Az #unb gombra kattintva az #unt elem megjelenítése vagy elrejtése történik, animációval 1000 milliszekundumos időtartamban.
- Az #godb gombra kattintva a #godt elem megjelenítése vagy elrejtése történik, animációval 1000 milliszekundumos időtartamban.
- Az #gmb gombra kattintva a #gmt elem megjelenítése vagy elrejtése történik, animációval 1000 milliszekundumos időtartamban.

Minden egyes eseménykezelő a .toggle() metódust használja az elemek megjelenítésének vagy elrejtésének vezérlésére, és az animáció időtartamát 1000 milliszekundumra állítja. Itt <p> tagokat változtatunk. Ezek tartalmazzák a rövid leírását a játékfejlesztő engine/környezeteknek.

Miskolc, 2024.

6.doc.html

Ez a weboldal a dokumentációját tartalmazza a projektnek, egy beágyazott dokumentummal. A dokumentum leírja hogy miket tartalmaznak az oldalak, milyen fájlok szerepelnek a projektben és hogy hogyan épülnek fel ezek a fájlok.

```
<div style="width: 100%; height: 1000px;">
    <embed src="RDC1L4.pdf" type="application/pdf" width="100%" height="100%"/>
</div>
```

Ez a kód egy beágyazott PDF dokumentumot jelenít meg egy HTML oldalon egy elemen belül.

- A elem stílusai beállítják annak szélességét 100%-ra és magasságát 1000px-re, tehát a beágyazott tartalom teljes szélességben és 1000 pixel magasságban jelenik meg az oldalon.
- Az <embed> elem beágyazza a PDF fájlt az oldalba. Az src attribútum meghatározza a beágyazandó fájl forrását (RDC1L4.pdf), a type attribútum pedig jelzi, hogy a beágyazandó tartalom PDF formátumban van. A width és height attribútumok 100%-os szélességet és magasságot állítanak be az elemre, így az elfoglalja a teljes elérhető területet a -ben.

Ez a kód tehát lehetővé teszi egy PDF fájl beágyazását az oldalba, és biztosítja annak megjelenítését a megadott méretű és stílusú -ben.

7.Szorgalmi

7.1.Server.js:

```
const http = require('http');
const fs = require('fs');
const path = require('path');
const cowsay = require('cowsay');
const output = cowsay.say({ text: 'Hi Krisztián!! :)' });
console.log(output);
const server = http.createServer((req, res) => {

    let filePath = '.' + req.url;
    if (filePath === './') {
        filePath = './index.html';
    }
    const extname = path.extname(filePath);
    let contentType = 'text/html';
    switch (extname) {
        case '.js':
            contentType = 'text/javascript';
            break;
        case '.css':
            contentType = 'text/css';
            break;
        case '.json':
            contentType = 'application/json';
            break;
        case '.png':
            contentType = 'image/png';
```

```
            break;
        case '.jpg':
            contentType = 'image/jpeg';
            break;
    }
    fs.readFile(filePath, (err, content) => {
        if (err) {
            if (err.code === 'ENOENT') {
                res.writeHead(404);
                res.end('File not found');
            } else {
                res.writeHead(500);
                res.end('Server error');
            }
        } else {
            res.writeHead(200, { 'Content-Type': contentType });
            res.end(content, 'utf-8');
        }
    });
});
const PORT = process.env.PORT || 3000;
server.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

A localhost: “3000”-es porton való megjelenítésre node js konfigurálására került sor. Köztük “npm” kódok kerültek beírásra a terminálba.
Elindításhoz “npm start” parancs beírása szükséges.

Ez a Node.js kód egy egyszerű webszervert hoz létre, amely statikus fájlokat szolgál ki a böngészőknek.

- Az első részben importáljuk a szükséges modulokat, mint például az http, fs, path és cowsay.

- A cowsay modul segítségével előállítanak egy üzenetet, amelyet az output változóba mentenek, majd kiírják a konzolra.
- Ezután létrehoznak egy HTTP szervert a http.createServer() függvénnyel, ami egy eseménykezelőt kap paraméterként. Az eseménykezelő akkor fut le, ha valaki egy kérést intéz a szerverhez.
- A kérés URL-jét egy filePath változóba mentik, majd megvizsgálják, hogy a kérés melyik fájlt kéri. Ha a kérés '/' URL-re érkezik, akkor alapértelmezettként az 'index.html' fájlt fogja betölteni.
- A fájlkiterjesztés alapján meghatározzák a fájl típusát (contentType), majd aszerint állítják be a megfelelő HTTP fejlécet.
- A fs.readFile() függvénnyel olvassák be a fájlt a megadott elérési útvonalról.
    - Ha a fájl nem található, 404-es hibaüzenetet küldenek vissza. Ha más hiba történik, akkor 500-as hibaüzenetet küldenek.
    - Ha a fájl beolvasása sikeres volt, akkor 200-as státusszal és a fájl tartalmával küldik vissza a kliensnek a választ.
- Végül a szerver egy meghatározott porton (alapértelmezetten a 3000-es porton) elindul, és kiírja a konzolra, hogy a szerver melyik porton fut.

7.2.Package.json tartalma:

```
{
  "name": "rdc1l4webtech",
  "version": "1.0.0",
  "description": "beadando",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.19.2"
  }
}
```

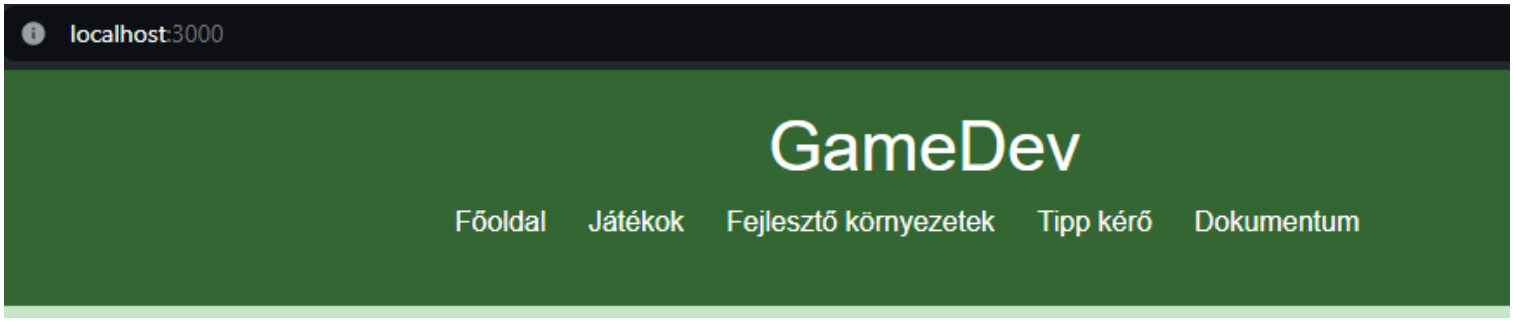
Ez egy package.json fájl, amely egy Node.js alkalmazás alapvető információit és konfigurációs beállításait tartalmazza. Néhány fontos részlet a fájlról:

- name: Az alkalmazás neve.
- version: Az alkalmazás verziószáma.
- description: Az alkalmazás rövid leírása.
- main: Az alkalmazás belépési pontja, itt található a fő fájl neve (ahol a szerver logika van), amely a server.js.
- scripts: A különböző parancsokat és scripteket tartalmazza, amelyeket a fejlesztés során lehet futtatni. Ebben az esetben csak egy start parancs van definiálva, amely a server.js fájlt futtatja a node server.js paranccsal.
- author: Az alkalmazás szerzője.
- license: Az alkalmazás licensze, itt az ISC (Internet Systems Consortium) licenc van megadva.

- dependencies: Az alkalmazás függőségei, amelyeket a projekt használ. Ebben az esetben csak az Express keretrendszer van megadva függőségként, a verziószám 4.19.2, ami azt jelenti, hogy a 4.19.2 verziótól kezdve a legújabb 4.x verzióig minden verziót elfogad. Ez azért fontos, hogy biztosítsa a kompatibilitást a különböző verziók között.

Ezt egy npm parancsal automatikusan létre lehet hozni.
A működéshez még lekellett tölteni a node.js-t és node modulokat.

Itt látható egy minta, hogy megjelenik az index.html, css-el is a localhost:3000-en.



8.Források:

https://freefrontend.com/css-cards/

https://getbootstrap.com/

https://www.w3schools.com/

https://en.wikipedia.org/wiki/List_of_video_game_genres

https://wall.alphacoders.com/

https://www.youtube.com/

Miskolc, 2024.