

**A következő dokumentum:**

**Jordán Kornél**

**Neptun: DWWUAP**

**Miskolci egyetem Programtervező  
informatikus hallgató**

**Webtechnológiák beadandóját  
mutatja be, egyszerű leírásokkal.**

# Index.html:



Az index.html egy olyan html oldal aminek a felépítési formáját a többi html-ben is követtem, ezért egyszer mutatom be.

```
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <meta charset="utf-8">
6     <title>Jordán Kornél webtech</title>
7     <link rel="stylesheet" href="css/design.css">
8     <script src="JavaScripts/html-include.js" type="text/javascript"></script>
9   </head>
10  <body class="bg">
11    <div class="parent">
12      <div class="div1">
13        <html-include src="header.html"></html-include>
14      </div>
15      <div class="div2">
16        <section class="middle cim">
17          <h1> Hadd mutakozzam be </h1>
18        </section>
19        <div class="middle">
20          
21          <section class="beginning">
22            <h3>Jordán Kornél</h3>
23            <h2>Harmadéves programtervező informatikus a Miskolci Egyetemen.</h2>
24          </section>
25        </div>
26      </div>
27      <div class="div3 smallmargin ">
28        <h1>Egy almafa <a href="secret.html" class="button2">.</a></h1>
29    </div>
```

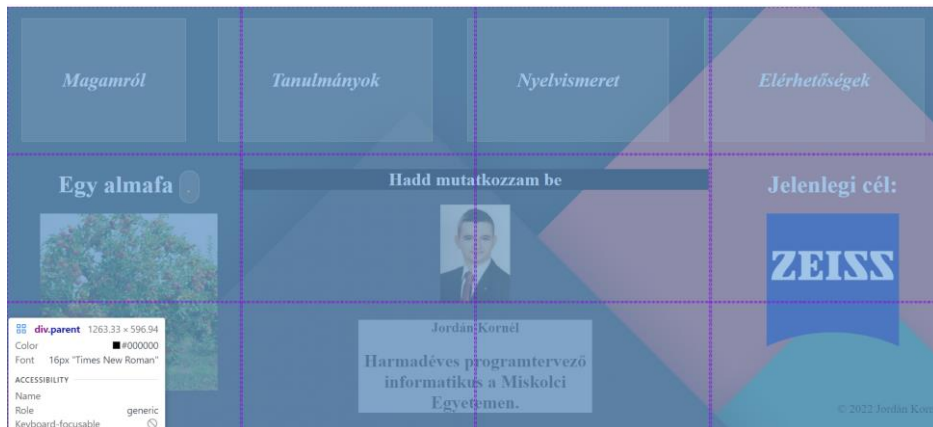
A css-ben megírt kinézeteket class-ok segítségével meghívom az adott részekhez és beszúrok képeket(src="img/en.jpg"), gombokat(<a href="secret.html">)amik más weboldalra mutatnak.

Ezeket a funkciókat ismételem és pozicionálom gridbox segítségével.

```
.parent {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-template-rows: repeat(3, 1fr);
  grid-column-gap: 3px;
  grid-row-gap: 2px;
}

.div1 { grid-area: 1 / 1 / 2 / 5; }
.div2 { grid-area: 2 / 2 / 5 / 4; }
.div3 { grid-area: 2 / 1 / 5 / 2; }
.div4 { grid-area: 2 / 4 / 5 / 5; }
```

A weboldal felbontása gridbox használata után így néz ki:



Most az oldalunk felső „címsávját” mutatom be:

```
<title>Jordán Kornél webtech</title>
<link rel="stylesheet" href="css/design.css">
<script src="JavaScripts/html-include.js" type="text/javascript"></script>
```

A link segítségével érem el a CSS fájlt amiben a grafikai megvalósítások vannak deklarálva.

A html-include.js egy olyan Javascript file amiben egyszer elkészített címsávot rajzoltatok ki minden olyan html-re amiben meg lett hívva. Így néz ki:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <header>
      <nav>
        <table class="table_header">
          <tr>
            <td class="button">
              <a href="index.html"><h1>Magamról</h1></a>
            </td>
            <td class="button">
              <a href="studies.html"><h1>Tanulmányok</h1></a>
            </td>
            <td class="button">
              <a href="languages.html"><h1>Nyelvismeret</h1></a>
            </td>
            <td class="button">
              <a href="contact.html"><h1>Elérhetőségek</h1></a>
            </td>
          </tr>
        </table>
      </nav>
    </header>
  </body>
</html>
```

Itt egyszerű <a href=""> segítségével linkeltem össze a html fileokat.

```

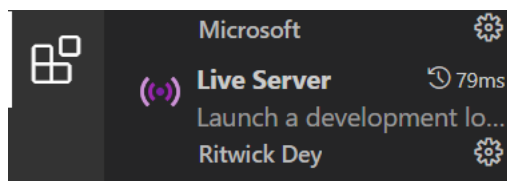
JavaScripts > JS html-include.js > ...
1  class HTMLInclude extends HTMLElement {
2      constructor() {
3          super();
4          this.innerHTML = "Loading...";
5          this.loadContent();
6      }
7
8      async loadContent() {
9          const source = this.getAttribute("src");
10         if (!source) {
11             throw new Error("No src attribute given.");
12         }
13         const response = await fetch(source);
14         if (response.status !== 200) {
15             throw new Error(`Could not load resource: ${source}`);
16         }
17         const content = await response.text();
18         this.innerHTML = content;
19     }
20 }
21
22 onDOMContentLoaded = (event) => {
23     window.customElements.define("html-include", HTMLInclude);
24 };
25 window.addEventListener("DOMContentLoaded", onDOMContentLoaded)

```

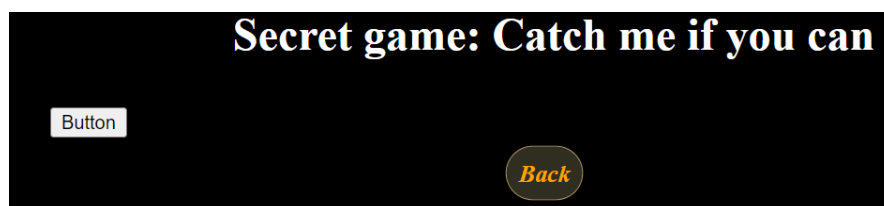
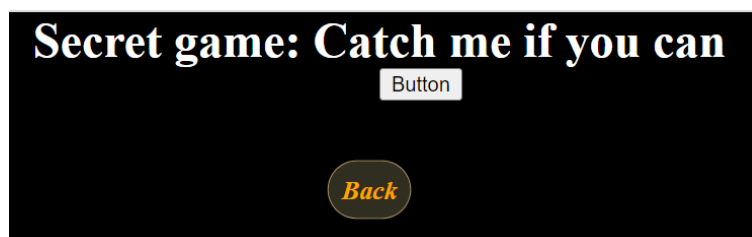
Itt először kiíratom, a „Loading...” kifejezést, amíg hiba nélkül be tudja olvasni a rendszer azt a file-t amit megadunk neki (hivatkozunk rá). Jelen esetben ezt így csináljuk:

```
<html-include src="header.html"></html-include>
```

Ahhoz, hogy működjön egy live serverként kell elindítanunk a weboldalt amit egy extension keretében tudjuk hozzá adni a Visual Studio Code-hoz.



**buttoneleporter.js:**



Még található egy játék az oldalunkon, ami egy gombra történő kattintás, azzal a nehezítéssel, hogy amikor rávisszük az egeret a gombra egy random helyre „átmegy”

```
JavaScripts > JS buttonteleporter.js > ...
1 let buttonteleporter = document.createElement("button");
2 buttonteleporter.innerHTML = "Button";
3 buttonteleporter.id = "buttontid";
4 document.body.appendChild(buttonteleporter);
5
6 const buttonHeight = 20;
7 const buttonWidth = 100;
8
9 const maxWidth = window.innerWidth - buttonWidth;
10 const maxHeight = window.innerHeight - buttonHeight;
11
12 window.addEventListener('DOMContentLoaded', () => {
13     const button = document.getElementById('buttontid');
14     button.addEventListener('click', () => alert("Winner"));
15
16     button.addEventListener('mouseover', () => {
17         button.style.left = Math.floor(Math.random() * (maxWidth * 1)) + 'px';
18         button.style.top = Math.floor(Math.random() * (maxHeight * 1)) + 'px';
19     })
20     buttonteleporter.style.position = "absolute";
21 })
```

Itt egy egyszerű Math.random() segítségével minden egyes alkalommal mikor rávisszük az egeret legenerálunk egy úgy koordinátát pixelbe és felülírjuk az előzőeket.

## word\_writer.js:



```

1  const word = "Ez a minta mert miért ne? ";
2  let wordChars = Array.from(word);
3  //document = new JSDOM(html).window.document;
4  document.querySelector("h1").innerHTML = word;
5
6  let btn = document.createElement("button");
7  btn.innerHTML = "Szöveg irány Fordító gomb";
8  btn.id="buttonid";
9
10
11 document.body.appendChild(btn);
12 let ok = true;
13 setInterval(function () {
14
15     if (ok==true) {
16         addEventListener
17         btn.style.left = "100px";
18         wordChars.push(wordChars.shift());
19         document.querySelector("h1").style.color = "red";
20     }
21     if(ok!=true){
22         wordChars.unshift(wordChars.pop());
23         document.querySelector("h1").style.color = "blue";
24         console.log(wordChars);
25     }

```

word\_writer.js:

Egy megadott szót ír ki, úgy hogy feldarabolja karakterekre és folyamatosan az elejére vagy a végére teszi az adott iránynak ellentétes oldalán lévő kezdő/végző karaktert. Ehhez, Javascriptben készítettem egy gombot amin egy addEventListener segítségével két állapot között változtatok. Az első állapotban egy push() segítségével „tolom” a szöveget és pirossá változtatom egy style.color segítségével. Másik állapotban unshift() segítségével pedig megfordítom az irányt és ugyanúgy egy style.color segítségével kékké változtatom a szöveget.

## form.js

A contact oldalon megtalálható egy form amivel adatokat tudunk felvinni.

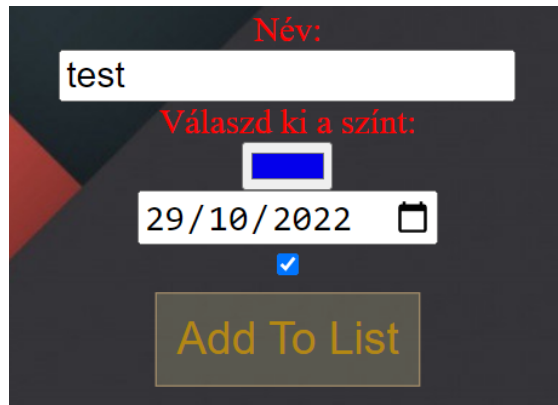
A név helyre be kell írunk valamilyen adatot, utána ki tudunk neki választani egy szint, dátumot és be kell pipálnunk a kis négyzetet, különben egy hiba üzenetet dob (alert).

127.0.0.1:5500 says

Nem fogadtad el!

OK

Abban az esetben, ha mindent jól töltöttünk ki:



Név:

test

Válaszd ki a szint:

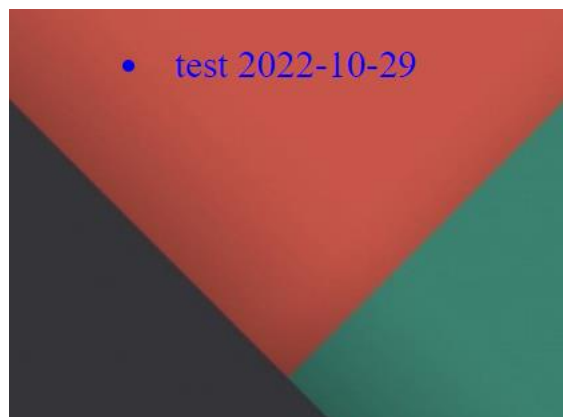
☐

29/10/2022 ☐

☒

Add To List

Felviszi a beírt adatot:



studies.html:

Ezen az oldalon a plusz rész ami bele lett téve, a felsorolás és egy videó a miskolci egyetemről ami loop-olva lett, hogy folyamatosan menjen, viszont ha szeretnénk le lehet állítani, ha rajta rákkattintunk a megfelelő gombra.



A loop részt a következő kódrészlettel oldottam meg:

```
<video controls loop width="480" height="300" autoplay>  
  <source src="mp4/University.mp4" type="video/mp4" >  
  <source src="mp4/University.ogg" type="video/ogg">  
  Your browser does not support the video tag.  
</video>
```

A listázást <li> segítségével lett megoldva:

```
<li>Miskolci Egyetem Gépészmérnöki és Informatikai kar</li>
```