

Web technológiák 1

Beadandó feladat

Bike Shop

Nyíri Levente

Neptunkód: F023QC

G3BIW

Tartalomjegyzék

Előszó.....	3
Header	4
Globális CSS	4
Global	4
Kártyák.....	7
Observer	9
Kezdőoldal	10
Carousel.....	10
Kártyák.....	14
Black Segment	14
Árak kártya.....	15
Árak.....	16
Táblázat formázása	19
Gyakori kérdések	21
Kapcsolat	24
Kapcsolatok kártya.....	24
Üzenet küldés	24
Közösségi média	26
Sticker	27
Lapok, gombok	27
Progress bar.....	31
Színválasztó.....	34
Bejelentkezés/Regisztráció	37
Regisztráció.....	37
Bejelentkezés.....	40
Végző.....	42

Előszó

A beadandóm témája egy bicikli boltnak a weboldala, amely kerékpárok és kerékpáros kiegészítők árusításával foglalkozik. A weboldal témájának a színe fekete és fehér lett, ezzel is próbáltam minél inkább letisztultabb és modern hatást kelteni.

A követelmények mellett próbáltam több dolgot is beletenni, amelyeket „low-code” környezetből ismertem. Ezek a következők:

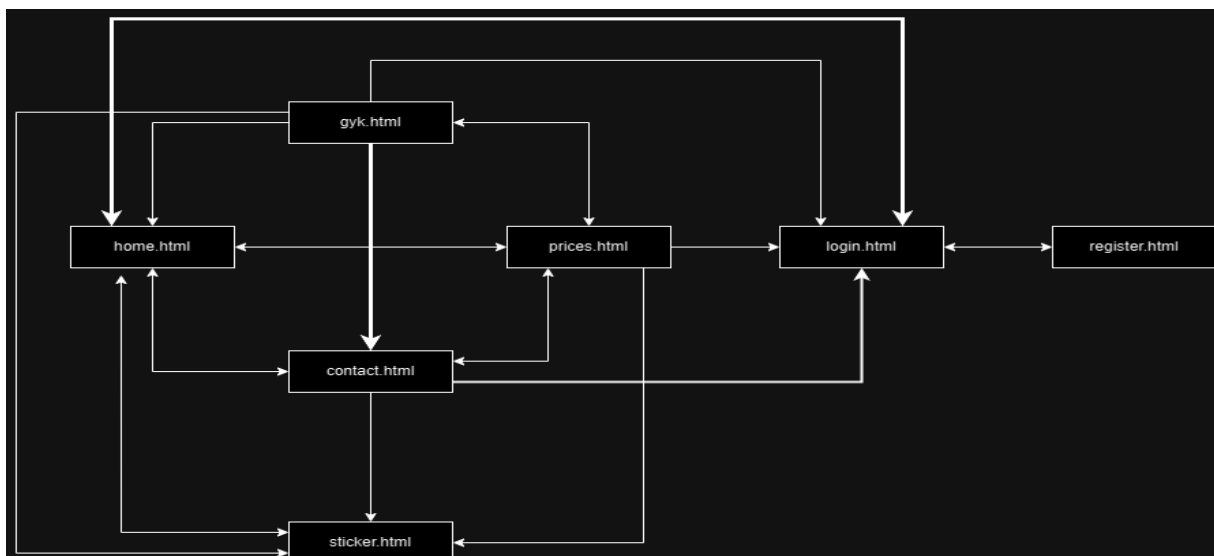
- Carousel
- Accordion
- Card

A weboldalak alapvető felépítése kettő darab HTML 5-ös elemből áll (header,main), azonban ez a felépítés a speciális oldalak esetében eltér. A header-t 3 részre lehet osztani:

- Logó
- Belépés/Regisztráció
- Menü

A main-nek a felépítése oldalanként változó, de leginkább 2 szekcióra bontottam fel, amelyek csak színükben térnek el (fekete és fehér). Ezeken a szekciókon belül a szövegek és a képek leginkább kártyákra helyezkednek el.

Összesen 7 darab html dokumentumot készítettem, amelyeket az alábbi kép mutat be:



Header

[Belépés/Regisztráció](#)[FREE Sticker](#)[Árak](#)[Kapcsolat](#)

A header-ben mindegyik elem mutat valahová. Ez a rész egységes az összes oldalon.

Az alábbi oldalakat lehet vele elérni:

Logóra kattintva – a főoldalra jutunk

Belépés/Regisztrációra kattintva a belépéshez jutunk

Free Sticker menüpontra kattintva pedig egy matricát tudunk igényelni

Árakra kattintva az árakat és a hozzá kapcsolódó információkat tekinthetjük meg

Kapcsolat gombra kattintva bejön a contact.html, ahol akár üzenetet is tudunk küldeni

Az egész header egy globális css fájl segítségével lett formázva, így bármikor létrehozhatunk egy új oldalt könnyedén.

Globális CSS

Összesen 2 darab CSS van, amit több oldalon is használtam ezek:

- global.css
- card.css

Global

Ebben a fájlban definiáltam a header és a main kinézetét, valamint általános beállításokat.

Általános beállítások:

```
*{
  margin: 0;
  padding: 0;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

body{
  overflow-x: hidden;
}

h1{
  text-align: center;
  text-transform: uppercase;
}
```

```
span{
  font-weight: bold;
}
```

A fejlesztés megkönnyítése érdekében mindenről leszedtem a margin-t és a padding-ot, valamint beállítottam, hogy milyen betűtípust szeretnék használni. Az overflow-x-et hiddenre állítottam.

A h1 és a span utólag került bele ebbe fájlba, mivel észrevettem, hogy túl sokszor írom be őket a többi CSS fájlba.

Header formázása:

```
header{
  display: flex;
  align-items: center;
  justify-content: center;
  position: fixed;
  top: 0;
  background-color: black;
  min-width: 100vw;
  max-height: 10vh;
  min-height: 10vh;
  color: white;
  z-index: 99999;
}

.logo{
  text-align: left;
  width: 50%;
  max-height: 10vh;
  padding-left: 15vw;
}

.links{
  width: 10%;
  text-align: right;
  padding-right: 5%;
  min-height: 100%;
}

.links a {
  font-size: 12px;
  color: white;
  font-weight: bold;
  text-decoration: none;
}

.links a:hover {
  font-size: 12px;
  color: white;
  font-weight: bold;
  text-decoration: underline;
  text-decoration: underline 3px;
}

.menu{
  width: 40%;
  display: flex;
}

.menu a{
  font-size: 20px;
  text-align: center;
  flex: 30%;
  color: white;
  font-weight: bold;
  text-decoration: none;
  background-color: black;
}

.menu a:hover{
  cursor: pointer;
  background-color: white;
  color: black;
  text-decoration: 2px underline;
  text-decoration: underline 3px;
  border-radius: 10px;
}
```

A headert úgy állítottam be, hogy folyamatosan a képernyőn maradjon. Fekete háttérszínt választottam neki és a benne lévő elemeket vízszintesen és függőlegesen is középre igazítottam. A headerben lévő 3 elemnek az alábbi szélességeket adtam: 50% 10% 40%, így kitöltik a rendelkezésre álló teret.

Mindegyik elem linkeket tartalmaz, kivéve a logót, ami egy képet. A linkeket megformáztam, illetve a menu-nak a linkjeire, ha rávisszük az egeret, akkor a fehér és a fekete szín megfordításával, illetve aláhúzással ki is emeltem, hogy éppen melyikre akarunk menni. A cursor: pointer az egeret változtatja meg.

Main:

```
main{
  margin-top: 10vh;
  width: 100%;
}

.section{
  justify-content: center;
  display: flex;
  align-items: center;
  margin-bottom: 10vh;
  width: 100%;
}

.black_segment{
  box-shadow: 0px 0px 58px 5vh rgba(0,0,0,1);
  margin-top: 20vh;
  padding-top: 5vh;
  margin-bottom: 20vh;
  padding-bottom: 5vh;
  width: 100%;
  background-color: black;
}
```

```
.white_segment{
  padding-top: 5vh;
  margin-bottom: 5vh;
  background-color: white;
  width: 100%
}
```

A mainre a margin top azért szükséges, mivel a header fix pozícióban van és 10 vh a magassága. Ha nem lenne ott, akkor a main elején lévő elemek bekerülnének a header alá, így nem lennének láthatóak. A main segmentekből épül fel, ezekben van benne a section. A két segment a háttérszínben különbözik, illetve az átmenetet box-shadow-val oldottam meg. Ezt amiatt tettem, hogy ne legyen túl hirtelen a váltás a két szín között.

Kártyák

Kétféle kártyatípust formáztam meg ebben az osztályban, amelyek méretükben és felépítésükben is különböznek.

Kis kártyák:

```
.card{
  flex: 30%;
  width: 30%;
  max-width: 30%;
  box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);
  border-radius: 5px;
  padding: 20px;
  background-color: white;
  height: auto;
  transition: transform 800ms ease-in;
}

.card:hover{
  box-shadow: 0 16px 32px 0 rgba(0,0,0,0.2);
}

.card_image{
  background-color: white;
  min-height: 30vh;
  height: 40%;
  border-radius: 5px;
}

.card_name{
  width: 100%;
  text-align: center;
  height: auto;
}
```

```
.card img{
  width: 100%;
}

.card ul,li{
  max-width: 100%;
  text-align: left;
  list-style-position: inside;
}

.card h2{
  margin-top: 10%;
  text-align: left;
  font-weight: bold;
  font-size: medium;
}
```

Ezek a classok vonatkoznak a kicsi kártyákra. HTML-ben pedig így néznek ki:

```
<div style="margin-right: 5%; margin-left: 5%;" class="card mid_c" id="mid_c">
  <div class="card_image">
    
  </div>

  <div class="card_name" >
    <h1>GIANT REIGN SX</h1>
    <p>Sok profi sportoló használja ezt a típust. Ha te akarsz lenni a leggyorsabb,
    <h2>Előnyök:</h2>
    <ul>
      <li>
        Nagyon megbízható
      </li>
      <li>
        Profi downhill felhasználás
      </li>
      <li>
        Profik által ajánlott
      </li>
    </ul>
  </div>
</div>
```

A kártyákon belül 2 div található. Az egyikben a képet a másikban pedig a szöveget tárolom. Minden kártyának van egy címe, ezt h2-vel oldottam meg. Egy felsorolást is ráaktam a kártyákra.

Nagy kártyák:

```
.big_card{
  width: 80%;
  max-width: 80%;
  box-shadow: 0 8px 16px 0 rgba(0, 0, 0, 0.2);
  transition: 0.3s;
  border-radius: 5px;
  background-color: white;
}

.big_card p {
  font-size: larger;
}

.big{
  width: 70%;
  min-height: 50vh;
}

.small{
  padding: 1%;
  width: 30%;
  min-height: 50vh;
  opacity: 0;
  transition: opacity 800ms ease-in;
}
```

A nagy kártyák 2 részből állnak:

- big
 - Ebben van a kép
- small
 - ebben van a szöveg

Ezeknek is vannak címük, illetve az egyik nagy kártya kép helyett iframe-et tartalmaz.

Observer

A kártyákhoz létrehoztam egy javascriptet, amellyel azt érem el, hogy görgetésre jönnek a szövegek a kártyákon, illetve a kis kártyákból 2 oldalról jön be a képbe.

```
const faders = document.querySelectorAll(".small");

const appearOptions = {
  threshold: 0.7
};

const appearOnScroll = new IntersectionObserver(function(
  smalls,
  appearOnScroll
) {
  smalls.forEach(small => {
    if (small.isIntersecting) {
      small.target.classList.add("appear");
      appearOnScroll.unobserve(small.target);
    }
  });
},
appearOptions);

faders.forEach(fader => {
  appearOnScroll.observe(fader);
});
```

Először kiszedem a dokumentumból az összes olyan elemet, amelynek az osztálya small.

Ezután megadom a beállításokat. A threshold: 0.7 azt jelenti, hogy a figyelt elem legalább 70%-ban kell látszódjon, mielőtt bejön képbe.

Ezután létrehoztam az intersection observert. Ennek a létrehozásához kell egy függvény és egy objektum, amiben a beállítások vannak. A függvénynek átadtam az objektumok listáját, amit figyelni akarok, és magát az observert. A függvény megnézi, hogy már látható-e az objektum, majd hozzáadja az appear osztályt. Végül pedig kiszedem azok listájából az objektumot, amelyeket figyelni kell, ezzel érem el azt, hogy csak egyszer jöjjenek be a kártyák, majd ha már bejöttek maradjanak is a helyükön.

Végezetül végig megyek az összes nagy kártyán és hozzáadom az observerhez.

A kis kártyák utólag kerültek bele a weboldalba, de ugyanazt az observert használtam.

```
const cards = document.querySelectorAll(".card");

cards.forEach(card => {
  appearOnScroll.observe(card);
});
```

Kezdőoldal

A kezdőoldal a következőképpen épül fel:

1. Header
2. Carousel
 - a. 3 oldalas
 - b. A középső oldal egy link, ami a RedBull oldalára mutat
3. White Segment
 - a. 3 darab kártya található meg benne
4. Black Segment
 - a. 2 darab nagy „kártya” van benne
 - b. Az egyikén beágyazott térkép
5. White Segment
 - a. 1 darab nagy kártya

Carousel

Ezt az elemet low-code környezetből ismertem meg. 3 oldala van és a lényege, hogy a nyilakkal tudjuk váltogatni, hogy melyik oldal jelenjen meg aktuálisan. Az oldalak az alábbiak:

1. oldal: A látogató köszöntése a weboldalon



2.oldal: Egy link, amely a RedBull oldalára visz



3.oldal: Egy motiválás



HTML:

```
<div class="carousel">
  <div class="slide">
    
    <h1 style="bottom: 5%;">Üdvözlünk a weboldalon!</h1>
  </div>
  <div class="slide">
    <a href="https://www.redbull.com/int-en/red-bull-hardline-2017-practice-gallery">
      
    </a>
  </div>
  <div class="slide">
    
    <h1 style="bottom: 5%; right: 5%; width: 30%;">"Cycling is more than a sport, it's a feeling of freedom and a journey of self-discovery."</h1>
  </div>
  <button class="c_button" id="prev" style="left: 5%;">&#8249</button>
  <button class="c_button" id="next" style="right: 5%;">&#8250</button>
</div>
```

A felépítését próbáltam minél egyszerűbbé tenni, illetve inline css-t használni a pozíciókra. A fő elem a carousel, ebben vannak benne a slideok, azaz oldalak és a gombok.

A gomboknál azért kell a left és right, hogy jobb és a bal oldalon legyenek.

```

.carousel{
  position: relative;
}
.slide{
  position: relative;
  display: none;
  max-height: 90vh;
}

.c_button {
  display: flex;
  justify-content: center;
  position: absolute;
  top: 50%;
  width: 50px;
  height: 50px;
  color: white;
  background-color: rgba(0, 0, 0, 0.3);
  border: none;
  border-radius: 50%;
  font-size: 40px;
  line-height: 40px;
  font-weight: bold;
}

.c_button:hover {
  background-color: black;
  cursor: pointer;
}

```

```

.slide img {
  position: relative;
  display: block;
  width: 100%;
  height: 100%;
  object-fit: cover;
  object-position: center;
}

.slide h1{
  font-size: 32px;
  position: absolute;
  color: white;
  text-align: center;
  width: 100%;
  text-shadow: -1px -1px 0 #000, 1px -1px 0 #000, -1px 1px 0 #000, 1px 1px 0 #000;
}

```


Magyarázat:

A gombokat úgy csináltam meg, hogy ne külön-külön kelljen definiálni. A display: flex és a justify-content: center azért kell, hogy a nyilak középre legyenek igazítva. border-radius:50% ezzel az érem el, hogy kör alakúak lesznek a gombok. A háttér segítségével, pedig elérem, hogy a gombok ki legyenek emelve, amikor kattinthatóak.

A háttérképet, hogy ne egyesével kelljen beállítani minden egyes slide-nál, beállítottam az img html element-et, hogy a töltsse ki a teljes teret, illetve középre igazítottam.

JavaScript:

```
const slides = document.getElementsByClassName("slide");
const nextButton = document.getElementById("next");
const backButton = document.getElementById("prev");
let currentPage = 0;
```

4 változót használtam, ebből 3 a html dokumentumból hivatkozik elemekre ID és Class segítségével. A slides jelöli az oldalakat, a kettő button pedig értelemszerűen a két gombot. A currentPage az aktuálisan aktív oldalt határozza meg.

```
function showPage(pageIndex) {
  for (let i = 0; i < slides.length; i++) {
    slides[i].style.display = "none";
  }
  slides[pageIndex].style.display="block"
}
```

A változók definiálása után létrehoztam egy függvényt, ami megkap egy számot és az alapján változtatja az oldalak megjelenítését. Először az összeset display: none-ra állítom egy for ciklus segítségével, majd az aktív oldalt blockra, ezzel elérem, hogy egyszerre csak egy oldal látható.


```
nextButton.addEventListener("click", () => {
  currentPage++;
  if(currentPage > slides.length-1){
    currentPage=0;
  }
  showPage(currentPage);
});

backButton.addEventListener("click", () => {
  currentPage--;
  if(currentPage < 0){
    currentPage=slides.length-1;
  }
  showPage(currentPage);
});
```

A két gombra esemény kezelőket raktam, ha megnyomják őket, akkor vagy hozzáadok egyet az aktív oldal számához, vagy kivonok belőle egyet. Az IF feltételek azért kellettek, hogy folyamatosan lehessen lapozni anélkül, hogy figyelni kelljen arra, hogy az utolsó vagy az első oldalon járunk. Végül pedig meghívom a fent említett függvényt.

Kártyák

LEGJOBB AJÁNLATOK



KTM PEAK

A legjobb választás mindenkinek, aki nem versenyszerűen szeretne biciklizni. Az aszfalton és az erdőben is megálja a helyét!

Előnyök:

- Olcsó
- Megbízható
- Ajánlott aszfaltnál és erdei utakhoz egyaránt



GIANT REIGN SX

sok profi sportoló használja ezt a típust. Ha te akarsz lenni a leggyorsabb, vagy egy megbízható, sok hibát engedő bringát keresel, akkor ez a legmegfelelőbb választás.

Előnyök:

- Nagyon megbízható
- Profi downhill felhasználás
- Profik által ajánlott



**GIANT TCR ADVANCED
DISC 1 KOM 2022**

Az utcai bringák királya. Ezt használva te lehetsz a leggyorsabb a csoport közül. Profik által ajánlott.


Előnyök:

- Carbon
- Kevés súly
- Profik által ajánlott

Az alapvető felépítésük:

- Kép
- Név
- Rövid ismertető
- Előnyök/jellemzők

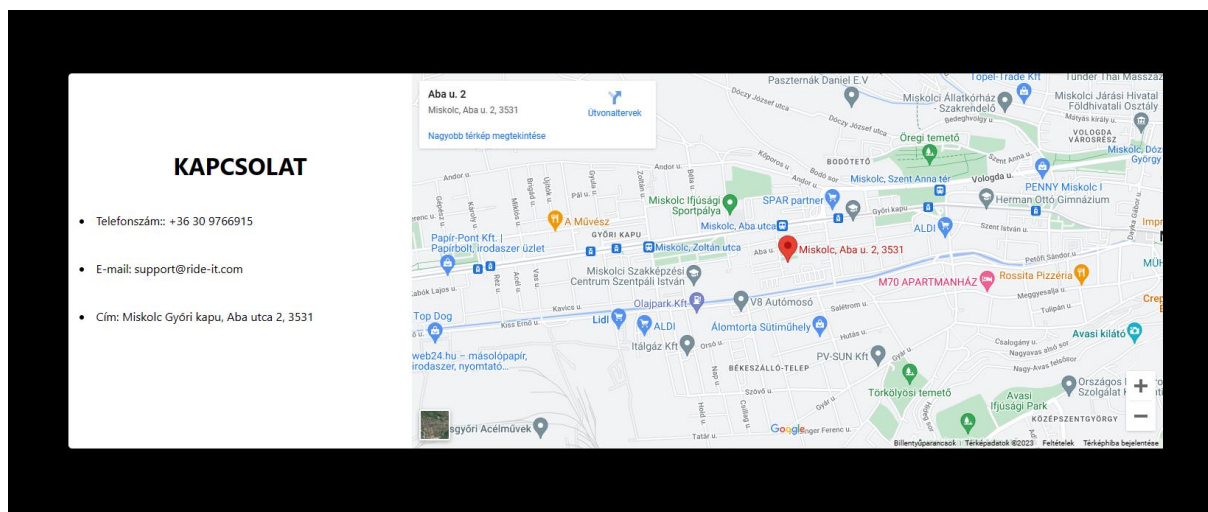
Black Segment



RÓLUNK

Üdvözlök a **Ride It!** weboldalamon, nálunk az összes szükséges kelléket megtalálod a biciklizéshez! A székhelyünk **Miskolcon** van. Nagyon elhívatottak vagyunk a bringázás iránt és célunk, hogy kezdőknek, illetve profiknak is egyaránt szolgáltatassunk felszerelést és eszközöket a szeretett sportjuk üzéséhez.

Kezdő vagy profi vagy? Nem számít. **Ha minket választasz megtalálod a legmegfelelőbb biciklit magadnak.**



Mindkét kártya, a már fent bemutatott HTML és CSS -el készült. A kapcsolatok kártyára egy térképet helyeztem el, ami mozgatható és nagyítható. Ennek a HTML kódja:

```
<iframe
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d2664.4012051869367!2d20.75274037702649!3d48.102495771239575!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x47409fc4448bb8e7%3A0x536f90b8a40aff79!2sMiskolc%2C%20Aba%20u.%20%2C%203531!5e0!3m2!1shu!2shu!4v1696278055243!5m2!1shu!2shu" width="100%" height="100%" style="border:0;"
allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade">

</iframe>
```

Árak kártya



Az utolsó kártya, amit használtam a kezdőlapra az az árakról szól, egy rövid szöveg van benne, illetve egy gomb, ami a prices.html-re mutat.

Árak

A prices.html-ben összesen 4 darab kártya van:



TERMÉKEK ÉS ÁRAK

Választékunkban számos kiváló minőségű termék található, azonban jelenleg nincs lehetőség a weboldalon keresztül vásárolni. A bolti árakat az oldal alján, egy táblázatban találhatja meg. Természetesen nem csak ezek a termékeink vannak, a boltban nagyobb választék van a biciklik és a kiegészítők között.

ÁRAJÁNLAT

Ha több terméket is szeretne venni nagy mennyiségben, akkor kérhet tőlünk árajánlatot. Ezt a kapcsolatok menüpont alatt tudja megtenni.



KEDVEZMÉNYEK ÉS AKCIÓK

Rendszeresen kínálunk különleges kedvezményeket és akciókat, hogy még vonzóbbá tegyük ajánlatainkat. Ne hagyja ki ezeket az alkalmakat! Kattintson a Kedvezmények oldalunkra, hogy részletesen megismerje aktuális ajánlatainkat.

GYAKORI KÉRDÉSEK

Ha további információra van szüksége az árakkal, a termékekkel vagy a szolgáltatásokkal kapcsolatban, kérjük, olvassa el **Gyakori Kérdéseinket** vagy lépjen kapcsolatba velünk. Válaszolunk minden kérdésére!



A gyakori kérdések kártyán van egy link, amelyet az alábbiak szerint formáztam:

```
a{  
  color: ☐black;  
  font-weight: bold;  
}
```

Az oldalon megtalálható még egy táblázat, amelynek a felépítése így néz ki:

PRICES			
Bike Name	Road/MTB	Price (HUF)	Professional/Non-Professional
Trek Domane SL6	Road	871,984	Professional
Specialized Rockhopper	MTB	186,863	Non-Professional
Giant TCR Advanced Pro 1	Road	984,000	Professional
Santa Cruz Bronson	MTB	1,299,463	Professional
Cannondale Synapse	Road	724,970	Non-Professional
Trek Fuel EX 8	MTB	1,067,856	Non-Professional
Pinarello Dogma F12	Road	3,600,000	Professional
Yeti SB130	MTB	1,487,978	Professional
Scott Addict RC 20	Road	1,054,583	Professional
Specialized Stumpjumper	MTB	1,119,938	Non-Professional

PRICES



Loading table...

```

<div style="display: flex; justify-content: center; margin-left: 5%; margin-right: 5%;">
  <table id="tabla" style="width: 100%; visibility: hidden;">
    <tr id="header_row">
      <td>Bike Name</td>
      <td>Road/MTB</td>
      <td>Price (HUF)</td>
      <td>Professional/Non-Professional</td>
    </tr>
    <tr>
      <td>Trek Domane SL6</td>
      <td>Road</td>
      <td>871,984</td>
      <td>Professional</td>
    </tr>
    <tr>
      <td>Specialized Rockhopper</td>
      <td>MTB</td>
      <td>186,863</td>
      <td>Non-Professional</td>
    </tr>
    <tr>
      <td>Giant TCR Advanced Pro 1</td>
      <td>Road</td>
      <td>984,000</td>
      <td>Professional</td>
    </tr>
    <tr>
      <td>Santa Cruz Bronson</td>
      <td>MTB</td>
      <td>1,299,463</td>
      <td>Professional</td>
    </tr>
    <tr>
      <td>Cannondale Synapse</td>
      <td>Road</td>
      <td>724,970</td>
      <td>Non-Professional</td>
    </tr>
    <tr>
      <td>Trek Fuel EX 8</td>
      <td>MTB</td>
      <td>1,067,856</td>
      <td>Non-Professional</td>
    </tr>
    <tr>
      <td>Pinarello Dogma F12</td>
      <td>Road</td>
      <td>3,600,000</td>
      <td>Professional</td>
    </tr>
  </table>
</div>

```

```

    </tr>
    <tr>
      <td>Yeti SB130</td>
      <td>MTB</td>
      <td>1,487,978</td>
      <td>Professional</td>
    </tr>
    <tr>
      <td>Scott Addict RC 20</td>
      <td>Road</td>
      <td>1,054,583</td>
      <td>Professional</td>
    </tr>
    <tr>
      <td>Specialized Stumpjumper</td>
      <td>MTB</td>
      <td>1,119,938</td>
      <td>Non-Professional</td>
    </tr>
  </table>
  <div id="load" style="position: absolute; font-weight: bold;">
    <div id="loader"></div>
    <p style="margin-top: 10px;">Loading table...</p>
  </div>
</div>

```

A táblázatot tároló div-ben megtalálható a táblázat, illetve egy töltési animáció.

Táblázat formázása

A táblázat és töltési animáció kinézetét <style> tagek, külön css fájl és javascript segítségével oldottam meg:

```

<style>
  table,td,tr{
    border: solid 1px black;
    border-collapse: collapse;
  }
  td{
    width: 25%;
  }

  @keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
  }

  #loader {
    border-radius: 50%;
    border-top: 16px solid black;
    border-right: 16px solid black;
    border-bottom: 16px solid black;
    border-left: 16px solid white;
    width: 10vh;
    height: 10vh;
    animation: spin 2s linear infinite;
  }
</style>

```

A border-collapse: collapse azért szükséges, hogy ne dupla vonallal legyenek határolva a táblázat mezői. A spin animáció biztosítja, hogy a töltési animáció mozogjon körbe. A töltéshez id-t használtam, amiben a border radius segítségével beállítottam, hogy a div az kör alakú legyen, illetve a borderek segítségével kiszedtem egy részét a körnek. Végül hozzáadtam az animációt, amelynek beállítottam az infinite segítségével, hogy folyamatosan menjen.

A külső CSS fájlban a táblázat legfelső sorát, a headert formáztam:

```
#header_row{
    background-color: black;
    color: white;
    border-color: black;
    font-weight: bold;
    text-align: center;
}
```

Végezetül pedig javascript segítségével beállítottam a töltési animációt, illetve formáztam a táblázatnak a sorait, hogy minden második sor szürke legyen.

```
var rows = document.getElementsByTagName("tr");

for (var i = 1; i < rows.length; i++) {
    if (i % 2 == 0) {
        rows[i].style.backgroundColor = "grey";
        rows[i].style.color="white";
    } else {
        rows[i].style.backgroundColor = "white";
    }
}

setTimeout(loaded,3000)

function loaded(){
    var loader = document.getElementById("load");
    var tablee = document.getElementById("tabla");
    loader.style.visibility = "hidden";
    tablee.style.visibility = "visible";
}
```

Gyakori kérdések

Ezen az oldalon használtam a low-code környezetből megismert accordion elemet. Ez az elem gyakorlatilag két egységből áll, az egyik az a cím, amely összezsukott állapotban is látható.

RIIDE IT!
Just Ride.

Beküldés/Regisztráció

FREE Sticker

Árak

Kapcsolat

GYAKORI KÉRDÉSEK

Van lehetőség online vásárolni?☐

Milyen fizetési módok vannak?☐

Van házhozszállítás?☐

Vissza lehet küldeni a terméket?☐

Mennyi a házhozszállítás költsége?☐

A weboldalon lévő árak aktuálisak?☐

Van lehetőség egyedi termék rendelésére?☐

Milyen típusú kerékpár lenne a legjobb választás számomra?☐

Milyen kerékpáros felszereléseket ajánlotok a biztonságos kerékpározáshoz?☐

Mikor érdemes kicserélni a bicikli gumikat?☐

A második pedig a cím alatt lévő rész, ahová bármit berakhatunk, lehet az kép, szöveg, iframe, stb...

Milyen fizetési módok vannak?

Bankkártya, készpénz, illetve átutalással lehet fizetni.

Van házhozszállítás?☐

Vissza lehet küldeni a terméket?

A terméket csak eredeti állapotában és csomagolásában, hiánytalan mennyiségi, minőségi állapotában vesszük vissza. A visszaküldés postaköltsége a vevőt terheli.

Az elem HTML kódja:

```
<div class="accordion">Van lehetőség online vásárolni?</div>
<div class="panel">
|   <p>Jelenleg sajnos csak személyesen lehet vásárolni, az üzletben.</p>
</div>
```

CSS:

```
.accordion {
  display: inline-block;
  background-color: white;
  color: black;
  cursor: pointer;
  padding: 18px;
  width: 80%;
  text-align: left;
  border: none;
  outline: none;
  box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);
}
```

```

border-radius: 5px;
margin-left: 10%;
margin-top: 1%;
}

.active, .accordion: hover, .accordion: hover::after {
background-color: black;
color: white;
font-weight: bold;
}

.panel {
display: inline-block;
background-color: white;
color: black;
padding: 18px;
width: 80%;
text-align: left;
transition: 0.4s;
box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);
border-radius: 5px;
margin-left: 10%;
display: none;
}

.accordion:after {
content: '\25EF';
font-size: 13px;
color: black;
float: right;
margin-left: 5px;
}

.active:after {
content: "\25C9";
color: white;
font-size: 18px;
}

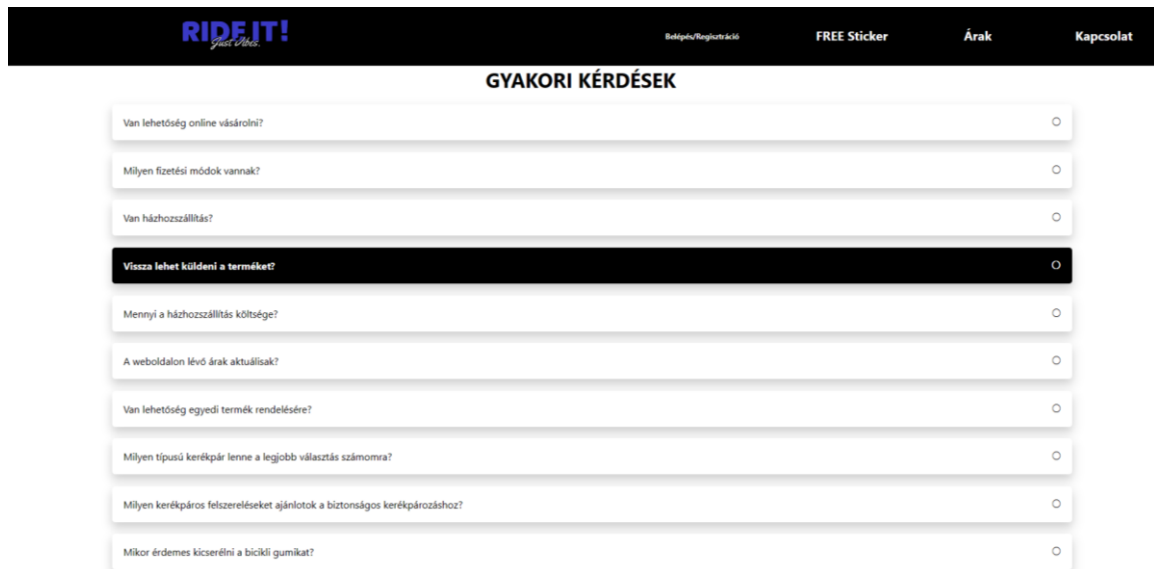
```

Az accordionnak 2 állapota lehet:

- Aktív
- Nem aktív

Az accordion akkor aktív, amikor rákattintunk és megjelenik a benne lévő tartalom. A :after segítségével állítottam be a kört, amely a jobb oldalán található az accordionnak. A contentnél meg tudjuk adni, hogy mi jelenjen meg az accordion után.

A hovererek esetében is beállítottam, hogy úgy nézzen ki, mintha aktív lenne, ezzel is kiemelve, hogy éppen melyikre fog rákattintani a felhasználó.



A javascript:

```
var acc = document.getElementsByClassName("accordion");

for (var i = 0; i < acc.length; i++) {
  acc[i].addEventListener("click", function() {
    this.classList.toggle("active");
    var panel = this.nextElementSibling;
    if (panel.style.display === "block") {
      panel.style.display = "none";
    } else {
      panel.style.display = "block";
    }
  });
}
```

Először az összes accordiont eltárolom egy változóban, majd egy for ciklus segítségével mindegyikhez hozzáadok egy eseménykezelőt. a toggle('active') segítségével tudom állítani, hogy az accordion aktív legyen-e vagy ne (ha aktív volt leszedelem róla a class-t, ha nem volt az, akkor hozzáadom). Ezután egy változóban tárolom a hozzátartozó panelt/contentet, amit a nextElementSibling segítségével kapom meg. Ez a funkció a html-ben keresi az ugyanolyan szinten lévő következő elemet. Végül egy if feltétel által eltüntetem, vagy megjelenítem a panelt.

Kapcsolat

Ezen az oldalon próbáltam meg bemutatni, hogy miért jó, a globális stílusok használata. Az oldal 3 elemből áll, amelyek közül 2 már látható volt a főoldalon. A 3 elem:

Kapcsolatok kártya



Ez az elem ugyanaz, mint a főoldalon. Ezzel bemutatva, hogy nagyon könnyedén újra felhasználhatjuk az elemeket.

Üzenet küldés

Az üzenet küldést a több soros beviteli mező miatt hoztam létre, ez is egy black segment-ben van benne, amit a főoldalon már láthattunk. Sajnos nem tárolom sehol az értékét, de van hozzá ellenőrzés, ahol azt ellenőrzöm, hogy a mező ki van-e töltve.

Alaphelyzetben:

ÜZENJ NEKÜNK!

Beküldés

Ha nincs kitöltve, de el szeretnénk küldeni:

ÜZENJ NEKÜNK!

Hiba! Az üzenet törzse üres!

Beküldés

Ha sikeresen elküldjük az üzenetet:

ÜZENET ELKÜLDVE!

A javascript hozzá:

```
function checkTextareaValue() {
    var textareaValue = document.getElementById('uzenet').value;

    if (textareaValue === null || textareaValue.trim() === '') {
        document.getElementById('uzenet').style.border = "3px solid red";
        document.getElementById("hiba").textContent = "Hiba! Az üzenet törzse üres!";
    } else {
        document.getElementById("uzi").style.display = "none";
        document.getElementById("done").style.display = "block";
    }
}
```

Egyszerűen ID alapján megkapom a beviteli mezőnek az értékét, majd egy if feltétellel megnézem, hogy üres-e. Ezután, ha nincs kitöltve megváltoztatom a stílusát, majd kiíratom, hogy hibás az érték. Ha nem üres a mező, akkor egyszerűen elrejttem a beviteli mezőt és megjeleníttem, hogy az üzenet el lett küldve.

HTML:

```
<div class="black_segment">
  <div class="section">
    <div class="big_card">
      <div id="done" style="display: none;"><h1>Üzenet elküldve!</h1></div>
      <div style="margin-left:5%; margin-top: 10vh; width: 90%; margin-bottom: 10vh; position: relative;" id="uzi">
        <h1>Üzenj nekünk!</h1>
        <form style="display: flex; justify-content: center; width: 100%;" action="">
          <textarea name="uzenet" id="uzenet" cols="30" rows="10"></textarea>
        </form>
        <p id="hiba" style="color: red; font-weight: bold;"></p>
        <div style="background-color: black; position: absolute; right: 0; color: white; margin-top: 5px; padding: 5px;" onclick="checkTextareaValue()">Beküldés</div>
      </div>
    </div>
  </div>
</div>
```

CSS:

```
textarea{
  padding: 1%;
  width: 100%;
  border: 3px solid black;
}

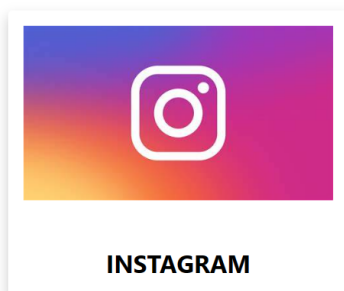
textarea:focus{
  outline: none;
}
```

Az `outline:none` azért kell, mert alapból ennek a típusú beviteli mezőnek van körvonala, amit nem szeretnénk látni, mivel a `border`-t már beállítottam.

Közösségi média

Ez a 3 kártya is a főoldalról lehet ismerős, ahol a legjobb ajánlatokat mutattam be. Kicsit átalakítottam, ezzel szemléltetve, hogy nem csak ugyanúgy nézhet ki az összes elem, ahogy a főoldalon van.

KÖZÖSSÉGI MÉDIA

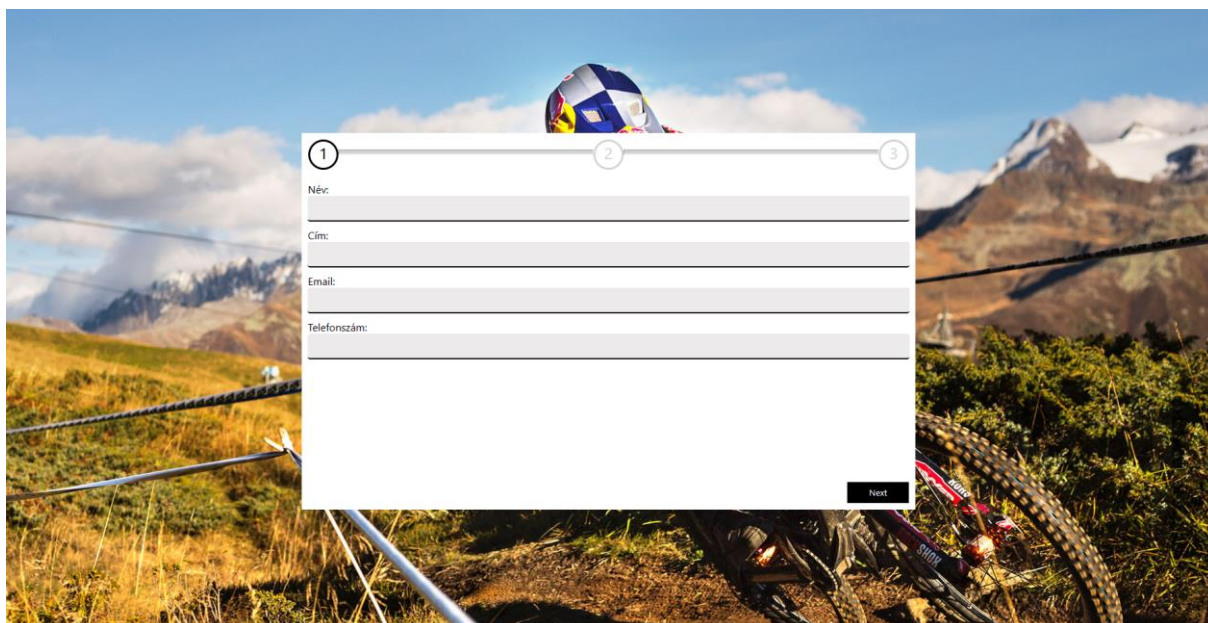


A kártyák kattinthatóak, és mindegyik a kártyának megfelelő linkre visz.

Sticker

Számomra ezt az oldalt volt a legnehezebb létrehozni, mivel még nem csináltam sohasem progress bar-t.

Az oldalon egy form található meg, amelyhez tartozik egy progress bar. Ezen az oldalon valósítottam meg a szín választót, illetve az input mezők ellenőrzését nem a szokványos „required” módszerrel oldottam meg, hanem egyesével ellenőriztem az értékeket.



Lapok, gombok

Összesen 3 gomb található meg az oldalon:

- Back
 - Az előző oldalra megy a formban
 - Első oldalon nem elérhető, mivel nincs előző oldal
- Next
 - Következő oldalra megy a formban
 - Utolsó oldalon nem elérhető, mivel nincs következő oldal
- Rendelés
 - Csak az utolsó oldalon található meg, így félkészben nem lehet leadni a formot

Az oldalak közötti lapozást a következőképpen oldottam meg:

```

function showPage(pageIndex) {
    progress(pageIndex);
    for (let i = 0; i < formPages.length; i++) {
        formPages[i].style.display = "none";
    }

    formPages[pageIndex].style.display="block"

    if(pageIndex == 0 ){
        backButton.style.display="none"
    }else{
        backButton.style.display="inline-block"
    }

    if(pageIndex == 2 ){
        nextButton.style.display="none"
        rendelesButton.style.display="inline-block"
        document.getElementById("nev_d").textContent= document.getElementById("nev").value;
        document.getElementById("email_d").textContent= document.getElementById("email").value;
        document.getElementById("cim_d").textContent= document.getElementById("cim").value;
        document.getElementById("tel_d").textContent= document.getElementById("tel").value;
    }else{
        nextButton.style.display="inline-block"
        rendelesButton.style.display="none"
    }
}
}

```

```

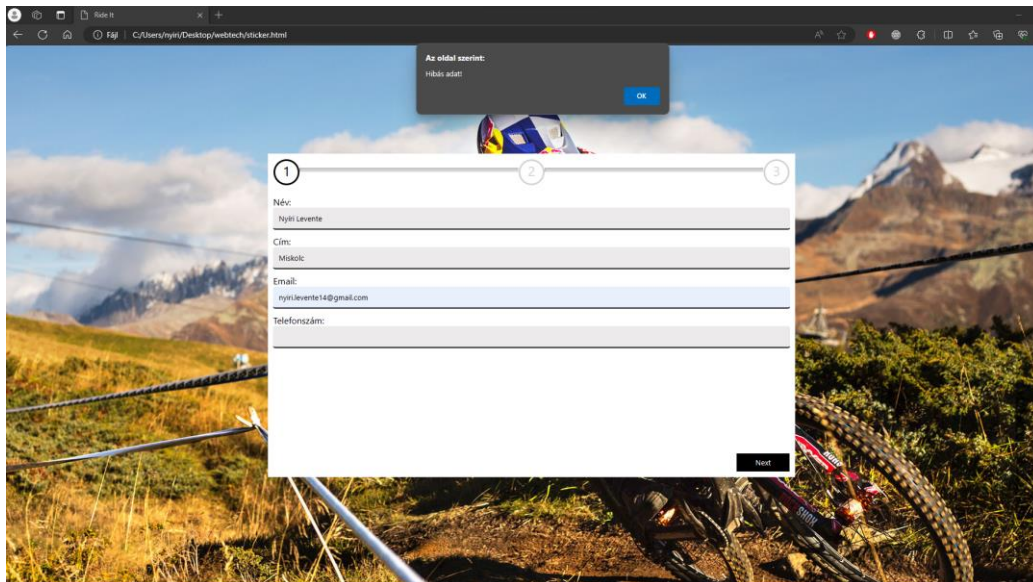
nextButton.addEventListener("click",() => {
    if(document.getElementById("nev").value != '' &&
document.getElementById("email").value != '' &&
document.getElementById("email").value.includes("@") &&
document.getElementById("cim").value != '' &&
document.getElementById("tel").value != ''){
        currentPage++;
        showPage(currentPage);
    }else{
        alert('Hibás adat!');
    }
});

backButton.addEventListener("click", () => {
    currentPage--;
    showPage(currentPage);
});

rendelesButton.addEventListener("click", () => {
    alert('Sikeres rendelés!');
    window.location.href = 'home.html';
});

```

A gombokra eseménykezelőket adtam hozzá. A next és a back button logikája az, hogy az aktuális oldalt egyel növeli vagy csökkenti és eszerint megjeleníti az adott oldalt. A Next gombnál ellenőrzöm, hogy ki vannak-e töltve a mezők, illetve hogy az email mező tartalmaz-e @-ot. Ha nincs, akkor egy üzenetet adok az alert segítségével, hogy hibás az adat, amelyet megadott. A rendelés gomb egy üzenetet ír ki, hogy Sikeres rendelés és visszamegy a főoldalra.



1. oldal

1

2

3

Név:

Nyíri Levente

Cím:

Miskolc

Email:

nyiri.levente14@gmail.com

Telefonszám:

06308566901

Next

Az első oldalon meg kell adnunk 4 adatunkat, mindegyik szöveges beviteli mező. Csak akkor léphetünk tovább, ha az összes ki van töltve és az email tartalmaz @-ot.


2. oldal

1

2

3

Válaszd ki a logót: Hard Line



Add meg a háttérszínt:

Red:

Green:

Blue:

Back Next

A második oldalon lehet kiválasztani a logót és a háttérnek a színét.

3. oldal

1

2

3



Rendelés adatai:

Név:
Nyíri Levente

Cím:
Miskolc

Email:
nyiri.levente14@gmail.com

Telefonszám:
06308566901

Back Rendelés

A harmadik oldalon Egy összegzőt kapunk, a megadott adatainkat és a kiválasztott matricát láthatjuk.

Progress bar

A progress bar-t azért hoztam létre, hogy a felhasználó láthassa, hogy a matrica igénylésének melyik részénél jár.

Összesen 3 állapot lehetséges:

- 
- Kész
- Aktuális
- Még nem kitöltött

A haladást pedig a következőképpen szimbolizáltam:



Látható, hogy az első lépés kész, a másodikon vagyunk jelenleg (aktuális) a 3. oldal pedig még kitöltésre vár.

```
function progress(pageindex) {  
  switch (pageindex) {  
    case 0:  
      document.getElementById("első").className="current";  
      document.getElementById("masodik").className="circle";  
      document.getElementById("harmadik").className="circle";  
  
      document.getElementById("line1").className="line";  
      document.getElementById("line2").className="line";  
      break;  
    case 1:  
      document.getElementById("első").className="done";  
      document.getElementById("masodik").className="current";  
      document.getElementById("harmadik").className="circle";  
  
      document.getElementById("line1").className="progress_line";  
      document.getElementById("line2").className="line";  
      break;  
    case 2:  
      document.getElementById("első").className="done";  
      document.getElementById("masodik").className="done";  
      document.getElementById("harmadik").className="current";  
  
      document.getElementById("line1").className="progress_line";  
      document.getElementById("line2").className="progress_line";  
      break;  
  }  
}
```

A javascriptet egy sima switch case segítségével oldottam meg. Az aktuális oldal szerint változtatom a köröknek az osztályát, így módosítom a stílusukat az adott oldal szerint.

HTML:

```
<div class="progress_bar">
  <div id="első">1</div>
  <hr id="line1">
  <div id="második">2</div>
  <hr id="line2">
  <div id="harmadik">3</div>
</div>
```

CSS:

```
.progress_bar {
  display: flex;
  justify-content: center;
  align-items: center;
  max-width: 100%;
  padding: 10px;
}
```

```
.current{
  border: solid;
  color: □black;
  min-width:40px;
  min-height: 40px;
  border-radius: 50%;
  line-height: 35px;
  display: inline-block;
  text-align: center;
  vertical-align: middle;
  font-size: 24px;
}

.circle {
  border: solid;
  color: ■lightgray;
  min-width:40px;
  min-height: 40px;
  border-radius: 50%;
  line-height: 35px;
  display: inline-block;
  text-align: center;
  vertical-align: middle;
  font-size: 24px;
}
```



```
.done{
  color: ■white;
  background-color: □black;
  border: solid;
  min-width:40px;
  min-height: 40px;
  border-radius: 50%;
  line-height: 35px;
  display: inline-block;
  text-align: center;
  vertical-align: middle;
  font-size: 24px;
}

.progress_line {
  width: 100%;
  height: 5px;
  background: □black;
  transform: translateY(-50%);
}

.line {
  width: 100%;
  height: 5px;
  background: ■lightgray;
  transform: translateY(-50%);
}
```

Színválasztó

A színválasztót a második oldalon oldottam meg slider-ek segítségével. Ezen az oldalon van még egy dropdown is, amivel a logót tudjuk módosítani.

HTML:

```
<div style="padding-left: 10px;" class="content">
  <div style="margin-bottom: 10px;">
    Add meg a háttérszínt:
  </div>

  <div>
    <p>Red:</p>
    <input type="range" min="0" max="255" value="0" class="slider" id="red">
  </div>
  <div>
    <p>Green:</p>
    <input type="range" min="0" max="255" value="0" class="slider" id="green">
  </div>
  <div>
    <p>Blue:</p>
    <input type="range" min="0" max="255" value="0" class="slider" id="blue">
  </div>
</div>
```

```
<div class="content">
  <div style="margin-bottom: 10px;">
    Válaszd ki a logót:
    <select id="logos">
      <option value="HardLine">Hard Line</option>
      <option value="Giant">Giant</option>
      <option value="KTM">KTM</option>
      <option value="RideIt">Ride It!</option>
    </select>
  </div>

  <div class="logoshow">
    
  </div>
</div>
```

CSS:

```
input[type="range"] {  
  -webkit-appearance: none;  
  -moz-appearance: none;  
  appearance: none;  
  width: 100%;  
  background-color: transparent;  
}  
  
input[type="range"]::-webkit-slider-runable-track {  
  -webkit-appearance: none;  
  appearance: none;  
  background: lightgray;  
  height: 0.5rem;  
}
```

```
input[type="range"]::-webkit-slider-thumb {  
  -webkit-appearance: none;  
  appearance: none;  
  border: 2px solid white;  
  border-radius: 50%;  
  background: black;  
  background-size: 100%;  
  box-shadow: 0px 3px 5px 0px rgba(0, 0, 0, 0.4);  
  width: 1rem;  
  height: 1rem;  
  cursor: grab;  
  position: relative;  
  bottom: 0.3rem;  
}  
  
input[type="range"]::-webkit-slider-thumb:active{  
  cursor: grabbing;  
}
```

Javascript:

```
logoDPD.addEventListener("change",()=>{
  switch (logoDPD.value) {
    case "HardLine":
      logo.src="sticker/Stickers/red-bull-hardline-logo.png";
      logo2.src="sticker/Stickers/red-bull-hardline-logo.png";
      break;
    case "Giant":
      logo.src="sticker/Stickers/giant.png";
      logo2.src="sticker/Stickers/giant.png";
      break;
    case "KTM":
      logo.src="sticker/Stickers/ktm-racing-1-logo-png-transparent.png";
      logo2.src="sticker/Stickers/ktm-racing-1-logo-png-transparent.png";
      break;
    case "RideIt":
      logo.src="sticker/Stickers/logo.jpg";
      logo2.src="sticker/Stickers/logo.jpg";
      break;
  }
});
```

A dropdownra raktam egy eseménykezelőt, ami a logókat változtatja. A logókat img elemekkel oldottam meg, emiatt elég csak a hivatkozásokat (src) megváltoztatnom a megfelelő képre.

```
var rangeInputs = document.querySelectorAll("input[type='range']");
rangeInputs.forEach(function(input) {
  input.addEventListener("change", changeBackground);
});

function changeBackground(){
  var red = document.getElementById("red").value;
  var green = document.getElementById("green").value;
  var blue = document.getElementById("blue").value;

  var logoshows= document.getElementsByClassName("logoshow");

  for (var i = 0; i < logoshows.length; i++) {
    logoshows[i].style.backgroundColor = `rgb(${red}, ${green}, ${blue})`;
  }
}
```

Ez pedig a div háttérszínének, vagyis a matrica hátterének a változtatása. 3 értéket lehet változtatni, ezért mind 3 slider-re ráraktam az eseménykezelőt, ugyanis így mind1 melyik változtatjuk, a háttérszín is változni fog mindhárom értéknek megfelelően. A for ciklus azért kell a végére, mivel folyamatosan változtatom a 2., illetve a 3. oldalon lévő logónak is a hátterét.

Bejelentkezés/Regisztráció

Ez a rész 2 HTML-ből áll, az egyik a regisztráció, a másik a bejelentkezés.

Regisztráció

Regisztráció

Felhasználónév

Jelszó

(legalább 8 karakter hosszú és tartalmaz nagy betűt)

Jelszó megerősítése

Születési dátum:

éééé. hh. nn.

Nem:

☐ Férfi ☐ Nő

☐ Elfogadom a [Felhasználói feltételeket](#).

Ha már van fiókod próbáld meg [bejelentkezni](#).

Regisztráció



Input típusok:

- szöveges
- jelszó
- dátum
- radio button
- checkbox

Az ellenőrzést a required paraméterekkel oldottam meg ezúttal.

Felhasználónév

Jelszó

(legalább 8 karakter hosszú és tartalmaz nagy betűt)

```
<input type="text" id="felhn" required>
```

Ezen kívül az a beépített ellenőrzésen kívül még a jelszavakat külön ellenőriztem:

```
if(/[A-Z]/.test(pwd) && pwd.length >= 8 && pwd===pwda){
```

Ezzel ellenőrzöm, hogy a jelszó tartalmaz-e nagy betűt, illetve a két jelszó, amit a felhasználó megadott egyezik-e. Ha a feltételeknek megfelel a két jelszó, akkor JSON-re alakítom majd eltárolom lokálisan. (Backend tudás hiányában csak így tudtam megoldani) Ezután pedig átváltok a bejelentkezés oldalra.

```
const logindata = {  
  username: felh,  
  pwd: pwd  
};  
localStorage.setItem('logindata', JSON.stringify(logindata));  
window.location.href = 'login.html';
```

```
document.getElementById('register').addEventListener('submit', function(event) {  
  event.preventDefault();  
  const felh = document.getElementById('felhn').value;  
  const pwd = document.getElementById('pwd').value;  
  const pwda = document.getElementById('pwda').value;  
  
  const prevError = document.getElementById('error-message');  
  if (prevError) {  
    prevError.remove();  
  }  
  
  if (/[A-Z]/.test(pwd) && pwd.length >= 8 && pwd === pwda) {  
    const logindata = {  
      username: felh,  
      pwd: pwd  
    };  
    localStorage.setItem('logindata', JSON.stringify(logindata));  
    window.location.href = 'login.html';  
  } else {  
    const errorMessage = document.createElement('div');  
    errorMessage.id = 'error-message';  
    errorMessage.className = "error"  
  
    if (!/[A-Z]/.test(pwd)) {  
      errorMessage.textContent = 'A jelszónak tartalmaznia kell legalább egy nagy betűt!';  
    } else if (pwd.length < 8) {  
      errorMessage.textContent = 'A jelszónak legalább 8 karakter hosszúnak kell lennie!';  
    } else {  
      errorMessage.textContent = 'A jelszavak nem egyeznek!';  
    }  
    document.body.appendChild(errorMessage);  
  }  
});
```

Ha nem felelnek meg a feltételeknek a jelszavak, akkor azt valamilyen módon jelezni kell. Ilyenkor egy új div-et hozok létre, aminek az osztályát beállítom errorra.

Az error class:

```
.error{
  background-color: red;
  color: white;
  position: absolute;
  top: 0;
  padding: 10px;
  text-align: center;
  width: 100%;
  z-index: 999;
}
```

Ezután a nem teljesített feltételnek megfelelő szöveget írom ki. Ennek tesztelése során felmerült az a probléma, hogy ha többször nem sikerül regisztrálni valamilyen feltétel miatt, akkor több error üzenet is a képernyőn marad. Ennek elkerülése miatt az alábbi kódot találtam ki:

```
const prevError = document.getElementById('error-message');
if (prevError) {
  prevError.remove();
}
```

Először megnézem, hogy van-e már hibaüzenet a képernyőn. Ha van akkor eltávolítom.

Az error üzenet így néz ki:

A jelszónak tartalmaznia kell legalább egy nagy betűt!

Regisztráció

Felhasználónév

Admin

Jelszó

(legalább 8 karakter hosszú és tartalmaz nagy betűt)

Jelszó megerősítése

Születési dátum:

2023. 10. 05.


Nem:

☒ Férfi ☐ Nő

☒ Elfogadom a [Felhasználói feltételeket](#).

Ha már van fiókod próbáld meg [bejelentkezni](#).

Regisztráció



Bejelentkezés

Bejelentkezés

Felhasználónév

Jelszó

Ha nincs fiókod [regisztrálj](#) először.



A jelentkezésnél szöveges, illetve jelszó típusú beviteli mezőket használtam. És itt is a fromot a required segítségével ellenőrzöm, hogy ki van e töltve, vagy nem.

HTML:

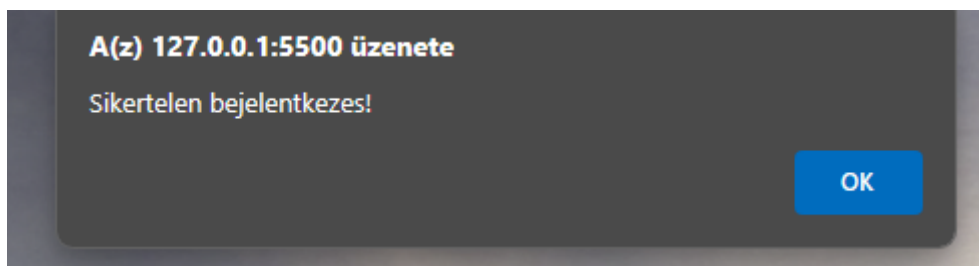
```
<div class="rightdiv">
  <form id="login" class="inputs">
    <h1>Bejelentkezés</h1>
    <p style="font-weight: bold;">Felhasználónév</p>
    <input type="text" id="felh" required>
    <p style="font-weight: bold;">Jelszó</p>
    <input type="password" id="pwd" required>
    <p>Ha nincs fiókod <a href="register.html">regisztrálj</a> először.</p>
    <button style="margin-top: 5%;" type="submit" >Bejelentkezés</button>
  </form>
</div>

<div class="videologin">
</div>
```


Javascript:

```
document.getElementById('login').addEventListener('submit', function(event) {  
    event.preventDefault();  
    const logindatac = JSON.parse(localStorage.getItem('logindata'));  
    const felhc = document.getElementById('felh').value;  
    const pwdc = document.getElementById('pwd').value;  
  
    if(felhc==logindatac.username && pwdc == logindatac.pwd){  
        alert('Sikeres bejelentkezés!')  
        window.location.href = 'home.html';  
    }else{  
        alert('Sikertelen bejelentkezés!')  
        document.getElementById('pwd').value = "";  
    }  
});
```

Először a lokális tárhelyből kieszedelem a felhasználói adatokat, majd a két input mező értékét. Ezután if-el ellenőrzöm, hogy megegyeznek-e az adatok. Itt az üzeneteket az alert segítségével iratom ki, nem hozok létre új elemet a html-ben. Ha sikeres a bejelentkezés akkor átnavigálom a felhasználót a kezdőoldalra, ha nem akkor pedig a jelszó mezőt üressé teszem.



Végszó

A feladat elkészítése körülbelül 5 héten át tartott, ezen időszak alatt próbáltam minden nap legalább 2 órát foglalkozni a feladattal. A megadott követelmények mellett a saját ötleteimmel is színesíteni szerettem volna a weboldalt.

Ami a legnagyobb fejtörést okozta a számomra, az a stickers.html-en lévő progress bar és a színválasztó csúszkáknak a megformázása volt.

Összességében nagyon tetszett a feladat, a követelmények teljesíthetőek voltak.