JEGYZŐKÖNYV

Web technológia alapjai

WatchOut karóra ismertető oldal

Készítette: **Szalóczy Krisztián**

Neptunkód: **Y4O4X0**

Dátum: 2024. május 11

**Miskolc, 2024**

**Tartalomjegyzék**

[Bevezetés 3](#_Toc166230577)

[1. Mappa struktúra 4](#_Toc166230578)

[1.1 Főoldal 4](#_Toc166230579)

[1.2 Videó megjelenítése 6](#_Toc166230580)

[1.3 Képek és információk megjelenítése táblázatban 7](#_Toc166230581)

[1.4 Űrlap elemek használata és validációja 7](#_Toc166230582)

[1.5 Az űrlap validációja JQuery használatával: 9](#_Toc166230583)

[2. Új div elem készítése: 9](#_Toc166230584)

[2.1 Rolex, Orient, Cartier, Baume oldalak ismertetése 11](#_Toc166230585)

[2.2 Navigációs sáv 11](#_Toc166230586)

[2.3 Image-slider megvalósítása 12](#_Toc166230587)

[2.4 A slider JavaScript implementációja 12](#_Toc166230588)

[3. Json fájlok és megjelenítésük az oldalon 14](#_Toc166230589)

[3.1 Ajax használata az adtok betöltéséhez 15](#_Toc166230590)

[3.2 Szorgalmi: node.js inicializálása, package.json, server.js, package-lock.json 15](#_Toc166230591)

# **Bevezetés**

Az általam készített projektben a karórák világát bemutató weboldal létrehozását választottam. Ez a döntésem arra vezethető vissza, hogy érdeklődéssel figyelem a divat és technológiai fejlődés találkozását, ami a karórák világában különösen izgalmasnak ígérkezik.

A karórák többek mint pusztán időmérő eszközök: egyúttal stílust és személyiséget is kifejeznek. Évszázadok óta hű társai az embereknek, és a technológiai előre lépesekkel párhuzamosan folyamatosan alakultak és fejlődtek. Ez a weboldal lehetőséget kínál arra, hogy átfogó képet kapjunk erről a fascináló világról, beleértve a történetüket, technikai jellemzőiket és a híres gyártóikat.

Az oldal célja, hogy könnyed, szórakoztató és informatív módon mutassa be a karórák világát. Szándékom, hogy a látogatókat magával ragadó élménnyel ajándékozzam meg, és egyben lehetőséget biztosítsak számukra arra, hogy mélyebben elmerüljenek ebben a témában. Így nem csupán egy egyszerű ismertető oldalt készítek, hanem egy olyan platformot, ahol a karórák szerelmesei felfedezhetik az újabbnál újabb karórákat.

A weboldal tervezése és fejlesztése során számos különböző technológia és eszköz kerül alkalmazásra annak érdekében, hogy egy modern és dinamikus felületet hozzak létre. Ennek részeként HTML-t és CSS-t használok az oldal struktúrájának és stílusának kialakításához, hogy a látogatók könnyen navigálhassanak és kellemes élményben részesüljenek az oldal böngészése során.

A felhasználói élmény fokozása érdekében beépítem a JavaScriptet és jQuery-t, amelyek segítségével interaktív elemeket és animációkat hozok létre az oldalon. Ez lehetővé teszi például a dinamikus tartalom betöltését, az interaktív űrlapokat vagy akár az animált effekteket, amelyek felhasználóbarát élményt biztosítanak.

# **1. Mappa struktúra**

Fő könyvtár: WebTechY4O4X0

* assets
* jquery
* watches
  + baume
  + cartier
  + orient
  + rolex
* index.html
* app.js
* server.js
* style.css

assest: A weboldalon felhasznált képeket és videókat tartalmazza.

jquery: A JQuery használatához szükséges JavaScript fájlt tartalmazza. A jelenlegi legfrissebb 3.7.1 verzió, amely letölthető az alábbi linken: <https://jquery.com/>

watches: További almappákat tartalmaz melyek a weboldalon megtekinthető html fájlok forrás kódját tartalmazzák , illetve a hozzájuk tartozó css, js, json fájlokat.

## **1.1 Főoldal**

A weboldal célja az általános információk megosztása az órák típusairól, müködésükről. A weboldal szerkezete tartalmazza a következő HTML elemeket: *article, section, aside, nav, header, footer.* Az órák típusairól szóló információk táblázatba rendezve jelennek meg amit *internal CSS*-el formáztam.

<nav>- tag használata a navigációs sávhoz.

<header>- tag használata a fejléc megjelítéséhez

|  |
| --- |
| **<nav** id="nav"**>**  **<header** class="main-title"**>**WatchOut**</header>**  **<hr>**  **<div** class="navbar"**>**  **<h3** class="menu-item"**>**Főoldal**</h3>**  **<div** class="dropdown"**>**  **<h3** class="menu-item"**>**Márkáink**</h3>**  **<div** class="dropdown-content"**>**  **<h3** class="brand"**><a** href="watches/rolex/rolex.html"**>**ROLEX**</a></h3>**  **<h3** class="brand"**><a** href="watches/orient/orient.html"**>**ORIENT**</a></h3>**  **<h3** class="brand"**><a** href="watches/cartier/cartier.html"**>**CARTIER**</a></h3>**  **<h3** class="brand"**><a** href="watches/baume/baume.html"**>**BAUME**</a></h3>**  **</div>**  **</div>**  **<h3** class="menu-item"**><a** href="#func"**>**Funkcionalitás**</a></h3>**  **<h3** class="menu-item"**><a** href="#mov"**>**Működés**</a></h3>**  **<h3** class="menu-item"**><a** href="#forras"**>**Források**</a></h3>**  **</div>**  **</nav>** |

<aside>- tag használata a videón levő szöveg megjelenítéséhez

|  |
| --- |
| **<aside** class="intro-text"**>**  **<span** class="intro-text-title"**>**WatchOut**</span>**   **<br><br>**  Csatlakozz hozzánk abban, hogy új korszakot nyissunk az óra gyűjtők számára. Fedezd fel helyszíneinket, globális közösségünket, gondosan összeállított kollekcióinkat - mind újakat, mind pedig gyűjteménybe tartozó előre használt darabokat - és beszéljünk órákról.  **</aside>** |

<article>-tag használat a bevezető szöveg megjelenítéséhez:

|  |
| --- |
| **<article** class="introduction"**>**  **<h1>**Üdvözöllek az Órák Világában!**</h1>**  **<p>**Ez az oldal bemutatja az órák funkcionalitását, és kiemeli a jelentős márkákat, mint például a Rolex, Orient, Cartier és Baum.**</p>**  **<p>**Az órák világa izgalmas és változatos. Minden időmérő mögött egyedi mechanizmus rejlik, amely precíz időmérésre és stílusos esztétikára képes.**</p>**  **<p>**Fedezd fel az alábbi táblázatot, hogy többet megtudj ezekről az órákról és működésükről!**</p>**  **</article>** |

<section>-tag használata szövegek megjelenítésére:

|  |
| --- |
| **<section** class="banner"**>**  **<div** class="banner-text"**>**  Fedezd fel az elegáns órák világát, ahol a mesteri kézművesség találkozik a kifinomultsággal minden részletben.**<br>**  **</div>**  **</section>** |

<footer>-tag használata a lábléc elkészítéséhez

|  |
| --- |
| **<footer>**  **<div** id="forras" class="footer-left"**>**  **<div** class="footer-menu"**>**  **<h1** class="footer-menu-title"**>**Források**</h1>**  **<ul** class="footer-menu-list"**>**  **<li><a** href="https://www.thewatchbox.com/"**>**https://www.thewatchbox.com/  **</a></li>**  **<li><a** href="https://www.thewatchpages.com/"**>**https://www.thewatchpages.com/**</a></li>**  **</ul>**  **</div>**  .....  **</footer>** |

## **1.2 Videó megjelenítése**

A navigációs sáv után egy rövid videót helyeztem el, a <video>-tag használatával, melynek *inline CSS-el* adtam meg a méreteit. A videón elhelyeztem egy gombot <button> aminek az onclick attribútumát használva meghívásra kerül egy Javascript függvény, amit az app.js tartalmaz. Ez a függvény elindítja/megállítja a videó lejátszását illetve változtatja a megjelenített ikont.

|  |
| --- |
| **<div** class="intro-container"**>**  **<div** style="width: 100%; height: 400px; overflow: hidden;"**>**  **<video** id="intro" src="/assets/intro.webm" style="width: 100%; height: auto;" controls**>**  **</video>**  **<button** id="toggleBtn" onclick="toggleVideo()"**>**  **<i** id="playIcon" class="fas fa-play"**></i>**  **<i** id="pauseIcon" class="fas fa-pause" style="display: none;"**></i>**  **</button>**  ....  **</div>** |

Az app.js *toggleVideo()* metódus implementációja:

|  |
| --- |
| **const** video = document.getElementById("intro");  **const** button = document.getElementById("toggleBtn");  **const** pauseIcon = document.getElementById("pauseIcon");  **const** playIcon = document.getElementById("playIcon");  **function** toggleVideo(){  **if** (video.paused) {  video.play();  playIcon.style.display = "none";  pauseIcon.style.display = "inline-block";  } **else** {  video.pause();  playIcon.style.display = "inline-block";  pauseIcon.style.display = "none";  }  } |

## **1.3 Képek és információk megjelenítése táblázatban**

Az órák csoportosítását funkcionalitás alapján egy táblázatba rendezve jelenítettem meg. A táblázat 2 soros <tr> és egy sorban 2 cella <td> - van amin a colspan attribútumot használva annyi helyet foglal el mint 2 db cella. Egy cellában egy <div>-en belül van egy kép az óráról és egy másik <div>-ben az adott óra típus információ. A táblázatnak internal CSS-el tehát a <head>-tagen belül <style> páros tag-et használva adtam stílust.

|  |
| --- |
| <div class="collection-container">  <div id="func" class="collection-title">Órák funkcionalitás szerint</div>  <hr>  <div class="table-container">  <table>  <tr>  <td>  <div class="content">  <img src="assets/analogFunc.png" alt="">  </div>  <div class="static-content-text">  <div class="static-content-top-left">Analóg</div>  <div class="static-content-middle"><span class="static-clock-feature">Kiemelkedő jellemző:</span> Az ...</div>  <div class="static-content-list">  <ul>  <li>Hagyományos óralap</li>  <li>Három mutató mozgás</li>  <li>Pénztárca barát</li>  </ul>  </div>  </div>  </td> |

## **1.4 Űrlap elemek használata és validációja**

Az űrlapot egy <div>-ben helyeztem el és egy azonosítóval id-val láttam el, amire későbbiekben fogok hivatkozni, mint css mint a JavaScript kódban. Az Űrlapon számos HTML elemet használtam a feladatnak megfelelően. A <form> -tagek között szerepelnek a feladatban megadott elemek, mint a szöveges beviteli mező egy/több soros, adatlista, jelölőnégyzet, rádió gomb, színválasztó, dátumválasztás, gombok.

|  |
| --- |
| <div id="popup-form">  <h2 id="urlap">Mi volt a tapasztalatod?</h2>  <form action="">  <label for="name">Név:</label>  <input type="text" id="name" name="name"><br>  <label for="name" class="error" id="error-name"></label><br>  <label for="email">E-mail:</label>  <input type="text" id="email" name="email"><br>  <label for="email" class="error" id="error-email"></label><br>    <label for="dateOB">Születési idő:</label>  <input type="date" id="dateOB" name="dateOB" min="1900-01-01" max="2025-01-01"><br><br>  <input type="radio" id="gender" name="gender">  <label for="gender">Férfi</label>    <input type="radio" id="female" name="gender" value="female">  <label for="female">Nő</label><br><br>  …. </div> |

Alapvetően maga az ürlap nem jelenik meg (display:none) az oldalon. Az oldalon létrehoztam egy gombot aminek az id-ra hivatkozva JQuery-t használva jelenítem meg az oldalon az űrlapot. A click eseményt meghívva a gombra nyomva előtűnik (fade) az űrlap. Illeteve a gombra is raktam egy JQuery animációt, amelynél a hover függvényt használtam ezáltal az egeret a gomb fölé helyezve megváltozik az áttünése (opacity) a gombnak.

|  |  |
| --- | --- |
| $(document).ready(**function**() {  $('#formBtn').hover(**function**() {  $(**this**).stop().animate({ opacity: **0.7** }, 'fast');  }, **function**() {  $(**this**).stop().animate({ opacity: **1** }, 'fast');  });  $("#formBtn").click(**function**() {  $("#popup-form").fadeIn();  });  $('#satisfaction').on('input', **function**() {  $('#satisfactionValue').text($(**this**).val());  });  $(document).mouseup(**function**(e) {  **let** container = $("#popup-form");  **if** (!container.is(e.target) && container.has(e.target).length === **0**) {  container.fadeOut();  }  }); | **#popup-form**{  **display**: **none**;  **position**: **fixed**;  **top**: **50**%;  **left**: **50**%;  transform: translate(**-50**%,-**50**%);  **background-color**: white;  **padding**: **20px**;  **border**: **2px** **solid** green;  **border**: **10px**;  box-shadow: **0px** **0px** **10px** rgba(**0**,**0**,**0**,**0**.**5**);  **z-index**: **1000**;  **width**: **300px**;  } |

## **1.5 Az űrlap validációja JQuery használatával:**

Az űrlap azonosítójára hivatkozva #popup-form JQuery-t használva megtudom hívni a submit esemény figyelőt. Amikor az űrlap beküldésre kerül, a **submit** eseményt megelőzően a kódrészlet meghívja az e.**preventDefault**() függvényt, hogy megakadályozza az alapértelmezett űrlap beküldési működését (oldal újratöltése). Ezután a kódban lévő input mezők értékeit ellenőrzi **(let name = $('#name').val();, let email = $('#email').val();).**

Ezután elvégzi a kötelező mezők (név és e-mail) validációját. Ha a név vagy az e-mail mező üres **(name === '', email === ''),** akkor piros szegéllyel jelez hibát **($('#name').css('border', '1px solid red');, $('#email').css('border', '1px solid red');),** és megjeleníti a megfelelő hibaüzenetet **($('#error-name').text('Name is required');, $('#error-email').text('Email is required');).** Ha az e-mail formátuma nem megfelelő (!isValidEmail(email)), szintén piros keretet és hibaüzenetet jelenít meg **($('#email').css('border', '1px solid red');, $('#error').text('Invalid email format');).**

|  |
| --- |
| $('#popup-form').submit(**function**(e) {  e.preventDefault();  $('#error').empty();  **let** name = $('#name').val();  **let** email = $('#email').val();  **if** (name === '') {  $('#name').css('border', '1px solid red');  $('#error-name').text('Name is required');  } **else** {  $('#name').css('border', '1px solid #ccc');  }  **if** (email === '') {  $('#email').css('border', '1px solid red');  $('#error-email').text('Email is required');  } **else** **if** (!isValidEmail(email)) {  $('#email').css('border', '1px solid red');  $('#error').text('Invalid email format');  } **else** {  $('#email').css('border', '1px solid #ccc');  }  }); |

# **2. Új div elem készítése:**

Az előzőekben bemutatott űrlapot kitöltve, a bevitt adatokat, ha azok megfelelőek akkor a submit gombra nyomva megjelenítem az oldalon. Amikor a dokumentum betöltődik (**'DOMContentLoaded'** esemény), az eseménykezelő függvény beállítja az űrlapot (**form**) és az adatok megjelenítésére szolgáló tartományt (**displayData**). Ezután hozzáad egy eseményfigyelőt az űrlap beküldéséhez (**form.addEventListener('submit', ...)**).

|  |
| --- |
| document.addEventListener('DOMContentLoaded', **function**(){  **let** form = document.getElementById('popup-form');  **let** displayData = document.getElementById("displayData");  form.addEventListener('submit', **function**(event) {  event.preventDefault();  **let** name = document.getElementById('name').value;  **let** email = document.getElementById('email').value;  **let** bornDate = document.getElementById('dateOB').value;  **let** gender = document.querySelector('input[name="gender"]:checked').value;  **let** satisfactionInput = document.getElementById('satisfaction');  satisfactionInput.addEventListener('input', **function**() {  **let** value = satisfactionInput.value;  document.getElementById('satisfactionValue').textContent = value;  });  **const** dataList = document.querySelector('#watch');  **const** options = dataList.querySelectorAll('option');  **let** favouriteWatches = [];  options.forEach(option => {  favouriteWatches.push(option.value);  }); |

Ezután létrehozok egy új **div** elemet (**datasDiv**), amelyben összeállítom az adatokat egy HTML formában, majd ezt az elemet hozzáadom a **displayData** elemhez, hogy megjelenítse az összegyűjtött adatokat. Végül az űrlapot elrejti (**form.style.display='none';**). A kód feltételezi, hogy minden szükséges mezőt megfelelően kitöltöttek. Ha valamelyik mező nincs kitöltve, a kód hibákat okozhat.

|  |
| --- |
| **let** datasDiv = document.createElement('div');  datasDiv.innerHTML = '<h3>Your Experiences</h3>'+  '<p><strong>Name:</strong> ' + name + '</p>'+  '<p><strong>E-mail:</strong> ' + email + '</p>' +  '<p><strong>Born date:</strong> ' + bornDate + '</p>' +  '<p><strong>Gender:</strong> ' + gender + '</p>' +  '<p><strong>Favourite watch:</strong> ' + favouriteWatches[0] + '</p>' +  '<p><strong>Satisfaction:</strong> '+ satisfactionInput.value + '</p>' +  '<p><strong>Message:</strong> ' + message + '</p>';  displayData.appendChild(datasDiv);  form.style.display='none'; |

## **2.1 Rolex, Orient, Cartier, Baume oldalak ismertetése**

Ezeknek a HTML oldalaknak a struktúrája egy forma. Mind az 5 oldalon látható a navigációs sáv, amin a linkek találhatóak. A Főoldal navigációs sávja különbözik a többitől, mert a márkáknak egy legördülő menüt alakítottam ki. Ebben a legördülő menüben találhatóak azok a linkek amik a további oldalakra vezetnek. A navigációs sávhoz a <nav>-taget használtam. A stílus kialakításához a CSS-be különböző szelektorokat használtam, mint példáúl osztály, azonosító, elem kijelölő.

|  |
| --- |
| **nav**{  **background-color**: black;  **color**: white;  **padding**: 20px 50px;  }  #navbar{  **color**: white;  }  **nav** **a**{  **text-decoration**: **none**;  **color**: white;  }  **.navbar**{  **display**: flex;  align-items: **center**;  **justify**-**content**: space-between;  } |

## **2.2 Navigációs sáv**

A navigációs sávot követően egy képek szúrtam be ami az oldal teljes szélességében fedi le. Magára az <img>-tagre használtam egy osztály (class) jelölőt amit felhasználva a JavaScript kódban JQuery-vel könnyedén eltudtam érni és használni rá a mouseenter metódust. Ezzel egy animációt adtam hozzá amivel megváltoztatom az áttünését (opacity). Ha rávisszük az egeret akkor 0.7 változik 300 millisec alatt, majd ha levesszük róla akkor vissza áll 1-re.

|  |
| --- |
| **<nav** id="nav"**>**  **<header** class="main-title"**>**WatchOut**</header>**  **<hr>**  **<div** class="navbar"**>**  **<h3** class="menu-item"**><a** href="/index.html"**>**FŐOLDAL**</a></h3>**  **<h3** class="menu-item"**><a** href="./rolex.html" target="\_blank"**>**ROLEX**</a></h3>**  **<h3** class="menu-item"**><a** href="/watches/orient/orient.html"**>**ORIENT**</a></h3>**  **<h3** class="menu-item"**><a** href="/watches/cartier/cartier.html"**>**CARTIER**</a></h3>**  **<h3** class="menu-item"**><a** href="/watches/baume/baume.html"**>**BAUME**</a></h3>**  **</div>**  **</nav>** |

|  |
| --- |
| $('.rolex-img-banner').mouseenter(**function**() {  $(**this**).stop().animate({ opacity: 0.7 }, 300);  }).mouseleave(**function**() {  $(**this**).stop().animate({ opacity: 1 }, 300);  }); |

A kép után egy szöveget helyeztem el egy div-ben, amit egyszerűen CSS- segítségével beállítottam az oldal közepére, sormagasságot, szint állítottam.

## **2.3 Image-slider megvalósítása**

Az oldalon elhelyeztem egy kép csúsztatót, amin adott az oldal témájának megfelelő óra márkájának 3 modellje látható. Ezek automatikusan váltják egymást, illetve az oldalán elhelyeztem két gombot, amivel a felhasználó is tovább pörgetheti a képeket. A <div>-ek egymásba vannak ágyazva ezzel lehet elérni a megfelelő pozíciót az oldalon és maguknak a képeknek a pozícióját a <div>-ken belül.

|  |
| --- |
| **<div** class="slider"**>**  **<div** class="slides"**>**  **<img** class="slide" src="/assets/rolex1.png" alt="rolex1"**>**  **<img** class="slide" src="/assets/rolex2.png" alt="rolex2"**>**  **<img** class="slide" src="/assets/rolex3.png" alt="rolex2"**>**  **</div>**  **<button** class="prevBtn" onclick="prevSlide()"**>**&#10094**</button>**  **<button** class="nextBtn" onclick="nextSlide()"**>**&#10095**</button>**  **</div>** |

## **2.4 A slider JavaScript implementációja**

A JavaScript kódban létrehoztam két konstanst amiből az egyiknek értékül adtam querySelectorAll() metódusban megadott elem összes gyerekének a képeit, ez a függvény egy nodeList-et ad vissza ami az oldalon található összes diaképet tartalmazza. A másik konstansban eltároltam azt az elemet ami tartalmazza a csúszka összes képét ez az elem lesz a diavetítés konténere.

|  |
| --- |
| **const** slides = document.querySelectorAll(".slides img");  **const** box = document.querySelector(".slides");  **let** slideIndex = 0;  **let** intervalId = **null**;  **let** models = []; |

**let slideIndex = 0;:** Ez a változó tárolja az aktuálisan megjelenített kép indexét a diavetítés során. Kezdetben ez az érték 0.

**let intervalId = null;:** Ez a változó tárolja az időzítő (timer) azonosítóját, amely a diavetítés automatikus előre léptetéséért felelős. Kezdetben ez az érték null.

**let models = [];:** Ez egy üres tömb, amelyet később használhatunk a diavetítéshez kapcsolódó modelladatok tárolására, például képaláírások, linkek vagy egyéb információk. Ez a tömb azonban jelenleg üres, és nem tartalmaz semmilyen adatot.

**let choosenModel = models[0];**: Ez a változó tárolja a jelenleg kiválasztott modellt az alapértelmezett modellként. Kezdetben ez az első modell a **models** tömbből.

**const currentModelImg = document.querySelector(".model-img");**, **const currentModelTitle = document.querySelector(".model-title");**, **const currentRelease = document.querySelector(".model-release");**, **const currentDescription = document.querySelector(".model-description");**: Ezek a változók tárolják azokat a HTML elemeket, amelyek a jelenlegi modell képét, címét, kiadási dátumát és leírását jelenítik meg.

A **slides.forEach((item, index) => { ... });** ciklus bejárja az összes diavetítési képet, és mindegyikhez hozzárendel egy eseménykezelőt, amely a képre kattintáskor fut le. Ennek eredményeként a kiválasztott modell a kattintott képhez lesz rendelve, és az adatai megjelennek az oldalon.

Az **initializeSlider()** függvény meghívásra kerül, amely beállítja az alapértelmezett diavetítést és elindítja az automatikus előre haladást. A **DOMContentLoaded** esemény figyelője biztosítja, hogy a diavetítés csak akkor induljon, amikor az összes tartalom betöltődött az oldalon.

|  |
| --- |
| **let** choosenModel = models[0];  **const** currentModelImg = document.querySelector(".model-img");  **const** currentModelTitle = document.querySelector(".model-title");  **const** currentRelease = document.querySelector(".model-release");  **const** currentDescription = document.querySelector(".model-description");  slides.forEach((item, index) => {  item.addEventListener("click", () => {  choosenModel = models[index];  currentModelTitle.textContent = choosenModel.title;  currentRelease.textContent = choosenModel.release;  currentModelImg.src = choosenModel.img;  currentDescription.textContent = choosenModel.description;  });  });  initializeSlider();  document.addEventListener("DOMContentLoaded",initializeSlider);  **function** initializeSlider(){  **if** (slides.length > 0) {  slides[slideIndex].classList.add("displaySlide");  intervalId = setInterval(nextSlide, 8000);  }  } |

**showSlide(index)**: Ez a függvény megjeleníti a megadott indexű diavetítési képet. Ellenőrzi, hogy az index nem lépi-e túl a diavetítési képek számát, és visszaállítja, ha igen. Ha az index negatív, akkor a legutolsó diavetítési képre ugrik. Ezután eltávolítja az összes diavetítési képet tartalmazó **.displaySlide** osztályt és hozzáadja azt az aktuális diavetítési képhez.

**prevSlide()**: Ez a függvény előreugrik egy diavetítési képpel az aktuális index alapján. Megállítja az automatikus előre haladást, csökkenti az indexet egyel, majd meghívja a **showSlide()** függvényt az új indexszel.

**nextSlide()**: Ez a függvény hátraugrik egy diavetítési képpel az aktuális index alapján. Növeli az indexet egyel, majd meghívja a **showSlide()** függvényt az új indexszel. Előtte azonban megszakítja az automatikus előre haladást, hogy ne zavarja a felhasználót.

A HTML-en belül létrehoztam két gombot amiknek az onclick attribútumába meghívom a prevSlide() és a nextSlide() JavaScript függvényeket a képek tovább pörgetésére.

|  |
| --- |
| **function** showSlide(index) {  **if** (index >= slides.length) {  slideIndex = 0;  } **else** **if** (index < 0) {  slideIndex = slides.length - 1;  }  slides.forEach(slide => {  slide.classList.remove("displaySlide");  });  slides[slideIndex].classList.add("displaySlide");  }  **function** prevSlide(){  clearInterval(intervalId);  slideIndex--;  showSlide(slideIndex);  }  **function** nextSlide(){  slideIndex++;  showSlide(slideIndex);  clearInterval(intervalId);  } |

# **3. Json fájlok és megjelenítésük az oldalon**

|  |
| --- |
| {  **"models"** : [  {  **"id"** : 1,  **"title"**:"Rolex - submariner",  **"release"** : 1954,  **"img"** : "/assets/rolex1.png",  **"description"**:"Az első Submariner 1954-ben debütált...."  },  {  **"id"**: 2,  **"title"**: "Rolex - GMT Master 2",  **"release"**: 2007,  **"img"**: "/assets/rolex2.png",  **"description"**: "Leiden, Hollandia - 2007. október 11.: Termékkép..."  },  {  **"id"**: 3,  **"title"**: "Rolex - submariner - Sea - Dweller",  **"release"**: 2001,  **"img"**: "/assets/rolex3.png",  **"description"**: "Az aranyat fénye és nemesessége ...."  }  ]  } |

## **3.1 Ajax használata az adtok betöltéséhez**

A weboldalakon szereplő óra márkák modelljeinek adatait egy model.json fájlban tárolom. Amit a feladatnak megfelelően egy Ajax kéréssel töltök be az oldalra. Két fajta módszert is használok az adatok beolvasására az egyik a $.ajax({…}) a másik egyszerűbb megoldás a $.getJSON() metódus hívás.Az url amelyre a kérés megy ebben az esetben a model.json fájlra mutat. A dataType a várt adat formátumot jelöli. A success egy függvény amely akkor fut le ha az AJAX kérés sikeresen végrehajtódik. Az itt megadott **data** paraméter tartalmazza a válaszként kapott adatokat. Ebben az esetben az adatokat egy **models** változóba mentem el.

**error**: Egy függvény, amely akkor fut le, ha az AJAX kérés hibával tér vissza. Ez az eset lehetőséget ad a hibakezelésre, és a felhasználó értesítésére a problémáról. Ez a függvény megkapja a hibát leíró paramétereket, mint például a HTTP státuszkódot és az üzenetet.

|  |  |
| --- | --- |
| $.ajax({  url: "model.json",  dataType: "json",  success: **function**(data) {  models = data.models;  },  error: **function**(xhr, status, error) {  console.error("AJAX hiba:", status, error);  }  }); | $.getJSON("model.json",**function**(data){  models = data.models;  }).fail(**function**(status, error) {  console.error("AJAX hiba:", status, error);  }); |

A **$.getJSON** függvény egy AJAX kérést indít a **model.json** fájl lekérése érdekében. Ha a kérés sikeres, a fájl tartalmát a **data** változóba menti, majd a **models** változóba menti az adatokban található modelleket. Ha a kérés nem sikerül, a **.fail** metódus segítségével hibakezelést végez, és kiírja a hibaüzenetet a konzolra.

## 

## **3.2 Szorgalmi: node.js inicializálása, package.json, server.js, package-lock.json**

Az **’npm** **init’** parancsot kiadtam terminálban, amivel egy új **Node**.**js** projektet inicializálok. Ezután a **Node** **Packege** **Manager** számos kérdést feltett a konfigurációval kapcsolatban. Ezek közé tartoznak például a projekt neve, verziószáma, leírása, fő fájl neve és egyéb metainformációk. Ennek eredménye ként létre jött a **package**.**json** ami a konfigurációt és a függőségeket fogja tartalmazni.

|  |
| --- |
| {  **"name"**: "webtechy4o4x0",  **"version"**: "1.0.0",  **"lockfileVersion"**: 3,  **"requires"**: **true**,  **"packages"**: {  **""**: {  **"name"**: "webtechy4o4x0",  **"version"**: "1.0.0",  **"license"**: "ISC",  **"dependencies"**: {  **"cowsay"**: "^1.6.0"  }  },  **"node\_modules/ansi-regex"**: {  **"version"**: "3.0.1",  **"resolved"**: "https://registry.npmjs.org/ansi-regex/-/ansi-regex-3.0.1.tgz",  **"integrity"**: "sha512-+O9Jct8wf++lXxxFc4hc8LsjaSq0HFzzL7cVsw8pRDIPdjKD2mT4ytDZlLuSBZ4cLKZFXIrMGO7DbQCtMJJMKw==",  **"engines"**: {  **"node"**: ">=4"  }  }, |

Az **npm** **install** parancsot kiadva létre jön a **package-lock.json**, ez a fájl pontosan rögzíti, hogy mely verziókban települtek le a függőségek, valamint azok függőségei és azok verziói.

|  |
| --- |
| {  **"name"**: "webtechy4o4x0",  **"version"**: "1.0.0",  **"description"**: "my first node.js server",  **"main"**: "app.js",  **"scripts"**: {  **"test"**: "ng test",  **"start"**: "node server.js"  },  **"repository"**: {  **"type"**: "git",  **"url"**: "git+https://github.com/szaloczy/Y4O4X0WebTechGyak.git"  },  **"author"**: "Szalóczy Krisztián",  **"license"**: "ISC",  **"bugs"**: {  **"url"**: "https://github.com/szaloczy/Y4O4X0WebTechGyak/issues"  },  **"homepage"**: "https://github.com/szaloczy/Y4O4X0WebTechGyak#readme",  **"dependencies"**: {  **"cowsay"**: "^1.6.0"  }  } |

Létrehoztam a **server**.**js** fájlt ami tartalmaz egy egyszerű **Node.js** szerveralkalmazást definiál, amely a beépített **http**, **fs** és **path** modulokat használja a szerver működtetéséhez és a statikus fájlok szolgáltatásához. Ezek a modulok részei a Node.js alaptelepítésnek, és nem szükségesek külön telepíteni őket az **npm**-en keresztül.

|  |
| --- |
| **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';  } |

A szerver létrehozásakor egy eseménykezelőt állít be, amely minden beérkező kérést kezel. A szerver első lépése az, hogy meghatározza a kérés URL-jéből származó fájl elérési útvonalát (**filePath**). Ha a kérés URL-je a gyökérre mutat, akkor az alapértelmezett **index.html** fájlt betölti.

|  |
| --- |
| **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**;  } |

Ebben a részletben a kód meghatározza a kiszolgáló által küldött fájlok típusát (**contentType**) az elérési útvonal kiterjesztése (**extname**) alapján. Az elérési útvonal kiterjesztése alapján a kód beállítja a megfelelő MIME típust a HTTP válasz **Content-Type** fejlécében.

|  |
| --- |
| 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');  }  }); |

Ebben a részletben a kód olvassa el a kért fájlt (**filePath**) a **fs.readFile** függvény segítségével. Amennyiben a fájl olvasása során hiba történik (**err**), a kód különböző válaszokat küld a kliensnek attól függően, hogy milyen hiba történt.

* Ha a fájl nem található (**'ENOENT'** hibakód), a kód 404-es státuszkóddal válaszol a "File not found" üzenettel.
* Ha bármilyen más hiba történik, a kód 500-as státuszkóddal válaszol a "Server error" üzenettel.
* Ha a fájl olvasása sikeres volt, a kód 200-as státuszkóddal válaszol, és visszaküldi a fájl tartalmát a megfelelő típussal (**contentType**).

A **node**\_**modules** mappa létrehozásához be kell húznom egy függőséget. Egy szimpla modult választottam a <https://www.npmjs.com/package/cowsay> oldaláról, aminek az a neve hogy **cowsay.** Az **npm** **install** **cowsay** paranccsal lehúztam ezt a modult a **node**.**js** projektembe.

Ha futtatom a **node** **server**.**js** paranccsal akkor kifogja rajzolni a konzolra a **cowsay**.**say** függvény által visszaadott értéket.

|  |
| --- |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **< Hi Krisztián!! :) >**  **\ ^\_\_^**  **\ (oo)\\_\_\_\_\_\_\_**  **(\_\_)\ )\/\**  **||----w |**  **|| ||**  **Server running on port 3000** |