**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ**

**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**КУРСОВ ПРОЕКТ**

Предмет: Въведение в скриптовите езици

Тема: Mcl60 3D

Ученик: Цветомир Дунев Научен ръководител:

Христо Чернокожев *Даниел Димитров*

Атанас Кънев

СОФИЯ

2 0 2 4

**Съдържание:**

1.Първа глава

1.1. HTML и Css

1.2. JavaScript

1.3. Spline

2. Втора глава

2.1. Изисквания към програмния продукт

2.1.1.Функционални изисквания

2.1.2.Нефункционални изисквания

2.2. Аргументиран избор на езика за програмиране и софтуерните средства

3.Трета глава

3.1. HTML файл

3.2. JS файл

4. Четвърта глава

**Увод:**

Проектът, който представяме, е уебсайт, посветен на болида MCL60 от Формула 1. Основната цел на този сайт е да предостави изчерпателна информация за този впечатляващ състезателен автомобил, като обединява различни мултимедийни и информационни ресурси. Сайтът е структуриран така, че да задоволи любопитството както на ентусиастите на Формула 1, така и на специалистите в областта.

Въведението в историята на болида предоставя контекст и представя MCL60 в рамките на състезателната програма на McLaren. Посетителите ще научат за дебюта, ключовите моменти и постижения, които са допринесли за създаването на този съвременен шедьовър.

Малко по-надолу посетителите ще забележат интерактивен 3D модел на болида MCL60, който им позволява да разгледат автомобила от всякакъв ъгъл и да се запознаят с неговите ключови компоненти и иновации. Това е уникална възможност за потребителите да се потопят във виртуалното изживяване и да усетят технологичното съвършенство на този болид.

Сайтът също така включва подробна техническа информация за MCL60, като представя спецификациите на двигателя, аеродинамичните решения и други ключови характеристики. Тази секция е предназначена за тези, които искат да разберат какво прави този болид толкова специален и конкурентоспособен на пистата.

Не на последно място, сайтът разполага с видео от представянето на MCL60, което предлага визуално и аудио изживяване, показващо автомобила в действие и представящо ключови личности от екипа на McLaren. Галерията със снимки допълва визуалното съдържание, предоставяйки висококачествени изображения от представянето на болида и снимки на колата с най-ключовите роли в отбора.

**Първа глава:**

**1.1. HTML и CSS**

HTML и CSS са основополагащите технологии за изграждане на уеб страници. HTML ще бъде използван за структуриране на съдържанието на сайта, като създаване на заглавия, параграфи, списъци, изображения и мултимедийни елементи. CSS ще се използва за стилизиране на тези елементи, като предоставя възможности за оформление, цветове, шрифтове и адаптивен дизайн, който гарантира добър външен вид на сайта на различни устройства и екрани.

**1.2. JavaScript**

JavaScript е динамичен програмен език, който ще бъде използван за добавяне на интерактивност към уебсайта. С помощта на JavaScript ще създадем интерактивен 3D модел на болида MCL60, който потребителите ще могат да разглеждат и манипулират. Освен това, JavaScript ще бъде използван за реализиране на различни функционалности, като анимации, динамично зареждане на съдържание и обработка на потребителски събития.

**1.3. Spline**

Spline е мощен инструмент за 3D моделиране, който използва сплайнове - математически криви, които се дефинират от контролни точки. Тези криви се използват за създаване на гладки и прецизни форми, което е особено полезно при моделирането на сложни обекти като болида MCL60. С използването на Spline, можем да създадем висококачествен и детайлен 3D модел, който предоставя реалистично визуално представяне на автомобила.

**Втора глава:**

**2.1 Изисквания към програмния продукт**

При създаването на уебсайт, посветен на болида MCL60 от Формула 1, се дефинират следните основни изисквания:

**2.1.1Функционални изисквания:**

* Включване на интерактивен 3D модел на болида MCL60.
* Показване на техническа информация за колата, включително спецификации на двигателя, аеродинамични елементи и системи за охлаждане.
* Вграждане на видео с представянето на болида.
* Галерия със снимки на болида.
* състезателните му постижения.
* Предоставяне на контактна информация и връзки към социалните мрежи.

**2.1.2Нефункционални изисквания:**

* Уебсайтът трябва да бъде отзивчив и да се адаптира към различни устройства и размери на екрана.
* Времето за зареждане на страниците трябва да бъде минимално.
* Уебсайтът трябва да бъде лесен за навигация и използване.

**2.2 Аргументиран избор на езика за програмиране и софтуерните средства**

За реализацията на уебсайта бяха избрани следните технологии:

* **HTML** (HyperText Markup Language) се използва за структуриране на съдържанието на уебсайта.
* **CSS** (Cascading Style Sheets) осигурява визуалното оформление и стилове на страниците.
* **JavaScript** добавя интерактивност и динамични функции към уебсайта, като например управлението на 3D модела
* **Spline:** Използван за създаване на интерактивния 3D модел на болида MCL60, който позволява на потребителите да разглеждат детайлите на колата в реално време.
* **Font Awesome:** Библиотека за икони, използвана за добавяне на визуални елементи като икони за социални мрежи и контакти.

**Трета глава:**

**3.1. HTML файл(showcase.html)**

****

**DOCTYPE декларация:**

Тази декларация указва на браузъра, че документът е написан в HTML5. Това е стандартната декларация за HTML5 и гарантира, че документът ще бъде интерпретиран правилно.

**Елемент <head>**

Секцията <head> съдържа метаданни за документа, като връзки към стилове, шрифтове, икони и други ресурси, необходими за правилното функциониране и визуализация на уеб страницата.

**Връзка към локален CSS файл**

Този ред добавя локален CSS файл (showcase.css), който съдържа специфични стилове за уебсайта.

**Връзка към външен CSS файл**

Този ред добавя външен CSS файл от библиотеката Font Awesome, който предоставя икони и шрифтове.

**Връзка към Font Awesome JavaScript**

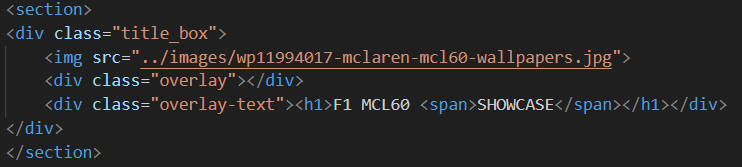
Този ред включва JavaScript файл от Font Awesome, който също предоставя допълнителни функционалности за иконите.

**Икона на сайта**

Този ред задава иконата на сайта, която ще се показва в таба на браузъра.

**Таг <title>**

Този ред задава заглавието на уеб страницата, което ще се показва в таба на браузъра.



**Елемент <section>**

Тагът <section> дефинира секция от документа, която групира съдържание по тематична свързаност.

**<div class="title\_box">**

Този елемент <div> с клас title\_box служи като групировка за всички елементи в заглавната част. Класът title\_box позволява лесно стилизиране на тази група чрез CSS.

**<img>**

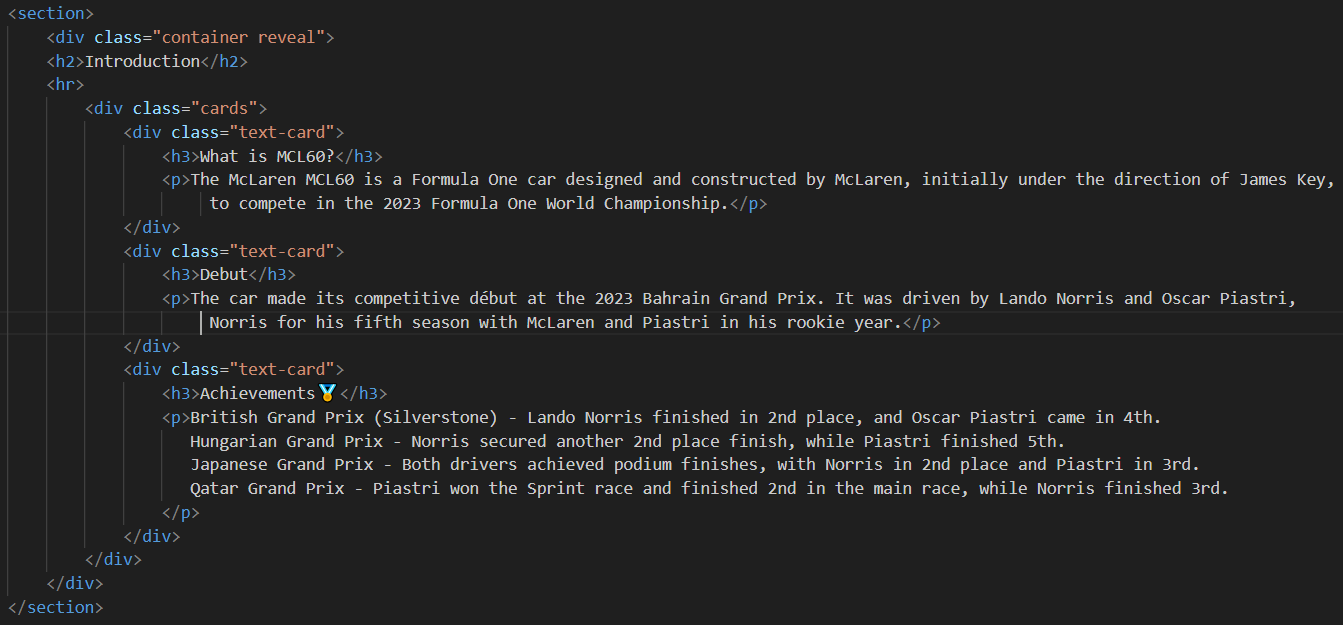
Тагът <img> включва изображение в документа. В този случай, изображението е на болида MCL60 и се намира в директорията ../images.

**<div class="overlay">**

Този елемент <div> с клас overlay представлява полупрозрачен слой, който се поставя върху изображението.

**<div class="overlay-text">**

Този елемент <div> с клас overlay-text съдържа текстовото съдържание, което се показва върху изображението. Вътре в него има заглавие <h1>.



**<div class="container reveal">**

Този елемент <div> с клас container reveal служи като главен контейнер за всички елементи в секцията "Introduction". Класът container се използва за подравняване и стил на съдържанието, докато класът reveal може да се използва за анимации при показване.

**<h2>**

Тагът <h2> дефинира заглавието на секцията, което в този случай е "Introduction".

**<hr>**

Тагът <hr> вмъква хоризонтална линия, която визуално разделя заглавието от следващото съдържание.

**<div class="cards">**

Този елемент <div> с клас cards служи като контейнер за група от текстови карти, всяка от които представя различна информация за болида MCL60.

**Текстова карта 1 <div class="text-card">**

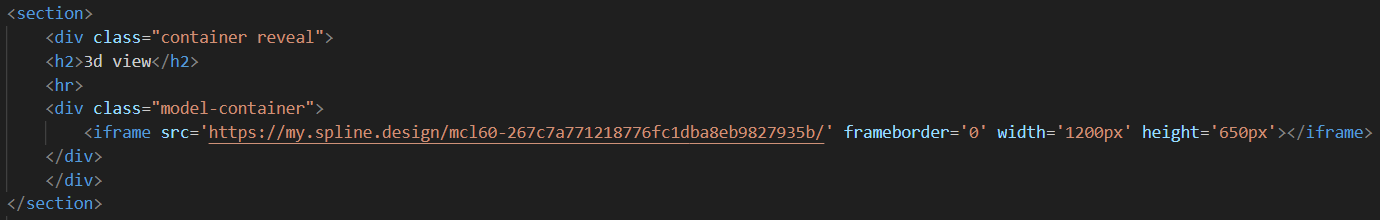
Този елемент съдържа заглавие <h3> и параграф <p>, които обясняват какво представлява болида MCL60.

**Текстова карта 2 <div class="text-card">**

Тази карта съдържа заглавие <h3> и параграф <p>, които предоставят информация за дебюта на болида MCL60.

**Текстова карта 3 <div class="text-card">**

Тази карта съдържа заглавие <h3> и параграф <p>, който изброява постиженията на болида MCL60 в различни състезания.



**<div class="container reveal">**

Този елемент <div> с клас container reveal служи като главен контейнер за всички елементи в секцията "3D View". Класът container се използва за подравняване и стил на съдържанието, докато класът reveal може да се използва за анимации при показване.

**<h2>**

Тагът <h2> дефинира заглавието на секцията, което в този случай е "3D View".

**<hr>**

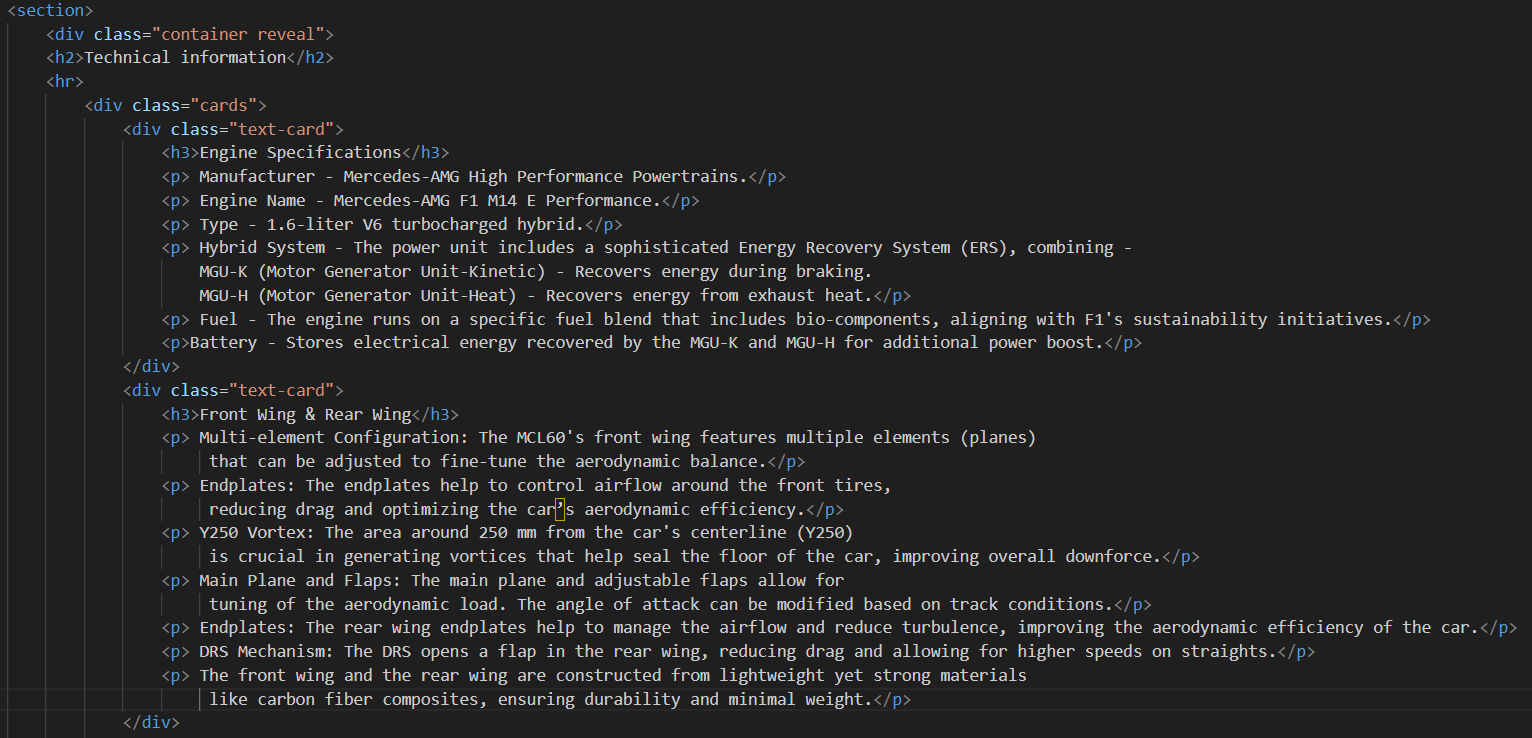
Тагът <hr> вмъква хоризонтална линия.

**<div class="model-container">**

Този елемент <div> с клас model-container служи като контейнер за 3D модела.

**<iframe>**

Тагът <iframe> вмъква интерактивен 3D модел в документа. Атрибутът src съдържа URL адреса към 3D модела, създаден със Spline. Атрибутите frameborder, width и height задават рамката на iframe-а и размерите му.



**<div class="container reveal">**

Този елемент <div> с клас container reveal служи като главен контейнер за всички елементи в секцията "Technical Information".

**<h2>**

Тагът <h2> дефинира заглавието на секцията, което в този случай е "Technical Information".

**<hr>**

Тагът <hr> вмъква хоризонтална линия.

**<div class="cards">**

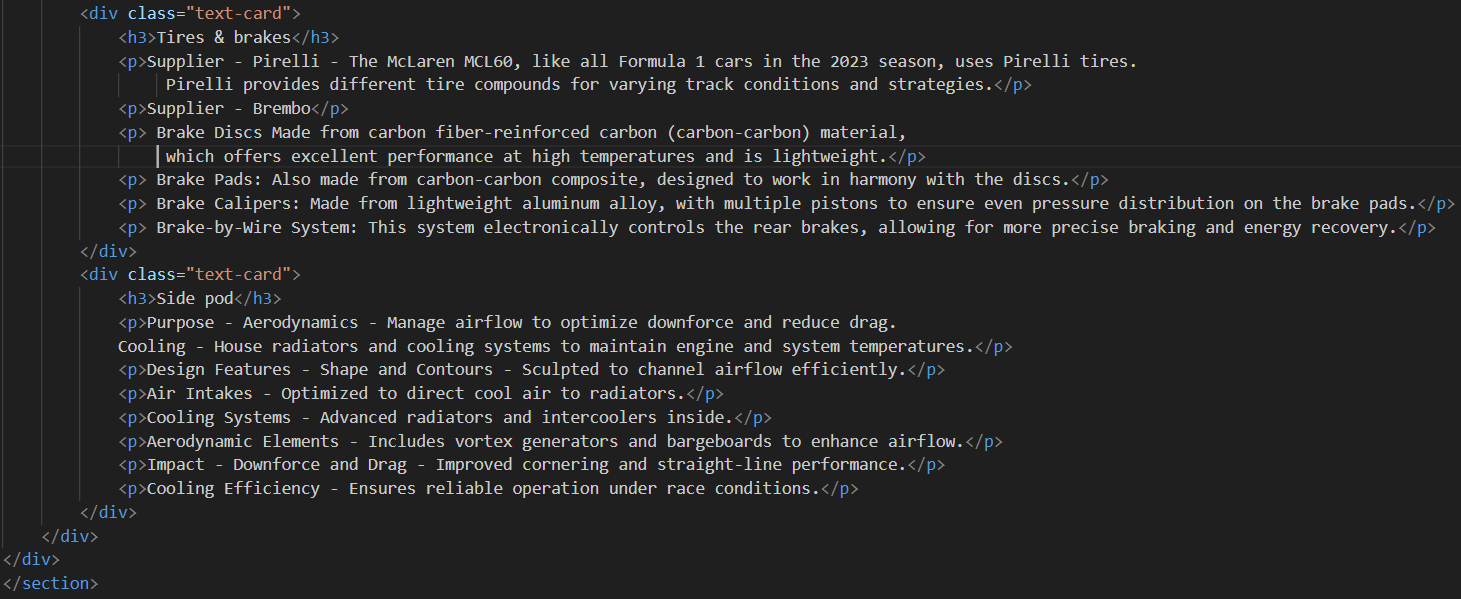
Този елемент <div> с клас cards служи като контейнер за група от текстови карти.

**Текстова карта 1 <div class="text-card">**

Тази текстова карта съдържа заглавие <h3> и параграфи <p>, които представят техническите спецификации на двигателя на болида.

**Текстова карта 2 <div class="text-card">**

Тази текстова карта съдържа заглавие <h3> и параграфи <p>, които представят информация за предното и задното крило на болида, както и за техните функции и конструкция.

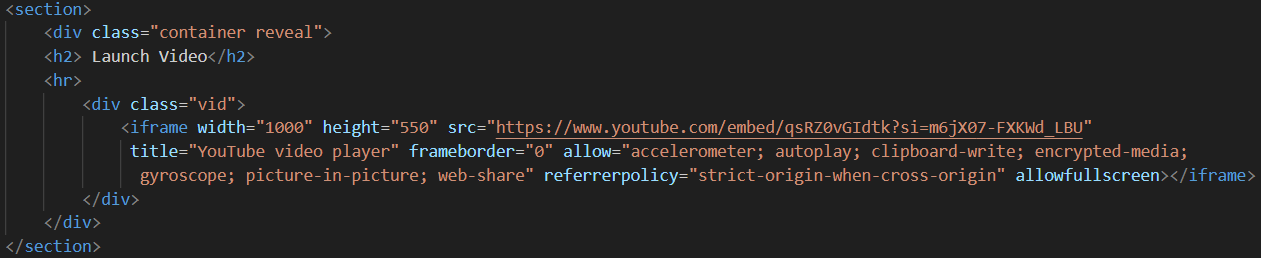


**Текстова карта 3 <div class="text-card">**

Тази текстова карта съдържа заглавие <h3> и параграфи <p>, които представят информация за спирачките и гумите на болида.

**Текстова карта 4 <div class="text-card">**

Тази текстова карта съдържа заглавие <h3> и параграфи <p>, които представят информация за страничните елементи на аеродинамиката.



**Заглавие**

Тагът <h2> съдържа заглавието на секцията "Launch Video".

**Видео елемент (iframe)**

Тагът<iframe> вмъква видео от външен източник, в случая от YouTube.



**Заглавие**

Тагът <h2> съдържа заглавието на секцията "Photo Gallery".

**Обвиващ контейнер за галерията**

Този <div> таг обгръща галерията и съдържа клас "menu-wrapper".

**Списък с изображения**

Тагът<ul> Задава неномериран списък със снимки. Всяко изображение е обгърнато от <li> таг с клас "item". Снимките са зададени с определени размери. Класовете "left-arrow", "right-arrow" и "arrow" задават стила и функционалността на бутона. Класът "hidden" прави бутона скрит по подразбиране. Изображението за стрелката е зададено с <img> таг.



**Бутон за скролиране към началото на страницата**

Таг <button> Създава бутон. id="scrollToTopBtn" – идентификаторът на бутона се използва за стилове и функционалност чрез CSS и JavaScript. Тагът <img> задава изображение в бутона, което представлява стрелка нагоре.

**футър**

Таг <footer> задава футъра секция на уеб страницата.

**Лявата част на футъра**

Таг <div> създава контейнер за лявата част на футъра. class="footer-left" – класът задава стиловете за левия контейнер във футъра.

**Заглавие на футъра**

Таг <h3> задава заглавие за футъра. Таг <span> обгръща текст "3D" в заглавието, което може да бъде използвано за стилови цели, като например оцветяване или форматиране на текста.

**Навигационни линкове във футъра**

Таг <p>: Създава параграф, който обгражда навигационните линкове.Клас на параграфа: class="footer-links" – класът задава стиловете за линковете във футъра. Тагове <a>: Задават линкове. Линковете водят до съответните секции или страници като "Blog", "Contact" и "About".



**Контактна информация**

Вътре в контейнера footer-center има няколко под-контейнера (допълнителни <div> елементи), всеки от които съдържа икона и текст с контактна информация. Таг<div> създава под-контейнер за всеки контакт.

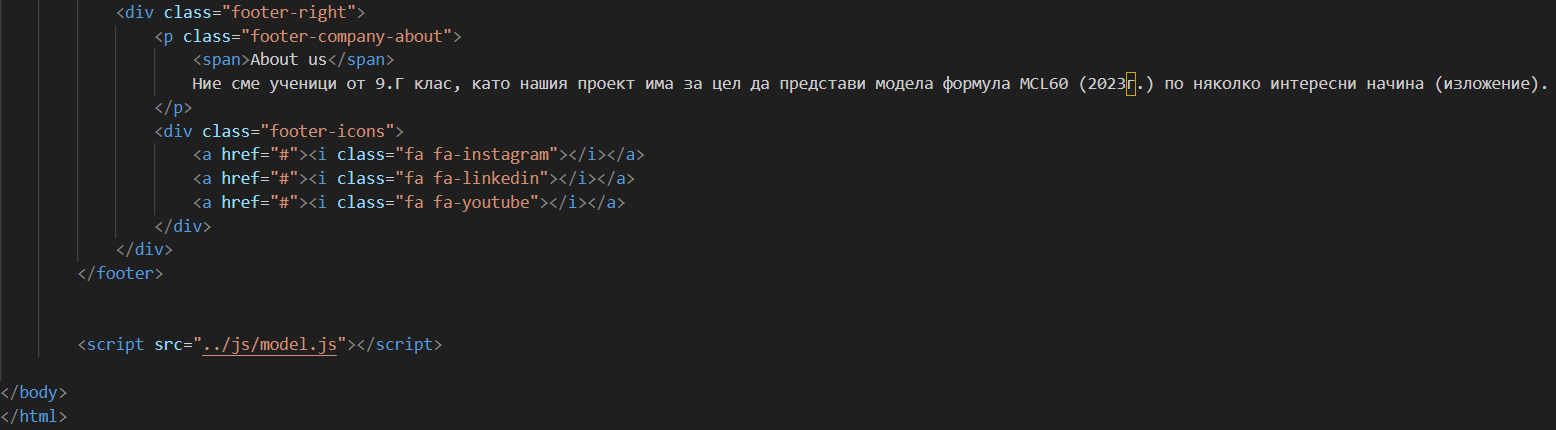
**Икона** **за** **телефон:**

Таг <i> се използва за добавяне на икона. Класове на иконата: class="fa fa-phone" – класовете са от библиотеката Font Awesome и задават стилове за иконата на телефон.

**Икона за имейл:**

Таг <i>: Използва се за добавяне на икона.

Класове на иконата: class="fa fa-envelope" – класовете са от библиотеката Font Awesome и задават стилове за иконата на имейл.



**About us**

Таг <p>: Създава параграф за текстовото съдържание. Клас на параграфа: class="footer-company-about" – класът задава стиловете за този параграф. Таг <span>: Обгръща текста "About us", което може да бъде използвано за стилови цели, като например оцветяване или форматиране на текста.

**Икони за социални мрежи**

Таг <div> създава контейнер за иконите на социалните мрежи.Клас на контейнера: class="footer-icons" – класът задава стиловете за този контейнер, което може да включва оформление и подравняване на иконите.

**Линкове към социални мрежи**

Таг<a>: Създава хипервръзка към съответната социална мрежа. В този случай, линковете водят до "#", което може да бъде заменено с реални URL адреси.

**Затварящи тагове за HTML документа**

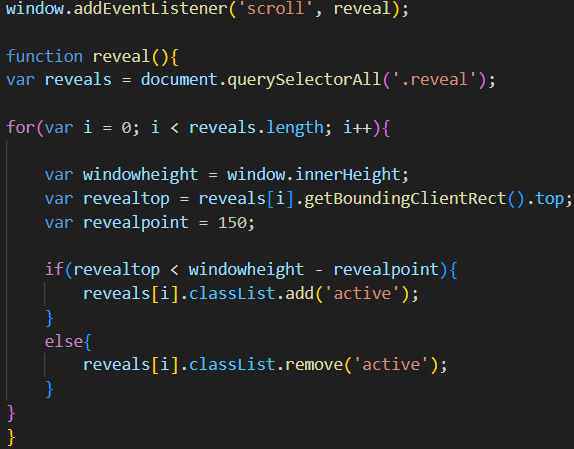
Затварящ таг за футър: </footer> – завършва футър секцията.

Таг <script>: Включва външен JavaScript файл, който се намира на „../js/model.js“.

Затварящ таг за тялото на документа: </body> – завършва тялото на HTML документа.

Затварящ таг за HTML документа: </html> – завършва HTML документа.

**3.2 JS файл**

****

**Добавяне на събитие за скролиране**

window.addEventListener('scroll', reveal): Това добавя слушател за събития към прозореца, който слуша за събитие scroll. Когато потребителят скролира, се изпълнява функцията reveal.

**Функция reveal**

Това е основната функция, която се изпълнява всеки път, когато потребителят скролира страницата.

**Избиране на всички елементи с клас reveal**

document.querySelectorAll('.reveal'): Това избира всички елементи в документа, които имат клас reveal, и ги съхранява в променливата reveals като NodeList.

**Цикъл за обхождане на всички reveal елементи**

Това е for цикъл, който обхожда всеки елемент в NodeList-а reveals.

**Взимане на височината на прозореца**

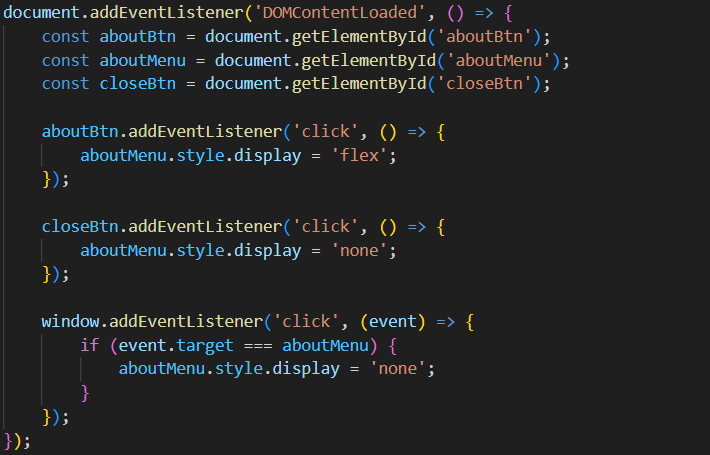
window.innerHeight: Взима текущата височина на прозореца на браузъра и я съхранява в променливата windowheight.

**Изчисляване на разстоянието от горната част на елемента до видимата част на прозореца**

reveals[i].getBoundingClientRect().top: Взима разстоянието от горната част на текущия елемент до горната част на видимата част на прозореца и го съхранява в променливата revealtop.

**Определяне на точката на разкриване**

revealpoint: Това е фиксирана стойност, която определя колко пиксела преди елементът да стане видим трябва да бъде разкрит. В този случай е 150 пиксела.



**Слушател за събитие DOMContentLoaded**

document.addEventListener('DOMContentLoaded', ...): Това гарантира, че скриптът ще се изпълни след като целият HTML документ е напълно зареден и анализиран. Това е полезно, за да се уверим, че всички елементи на страницата са налични в DOM преди да се опитаме да работим с тях.

**Избиране на елементи от DOM**

document.getElementById('aboutBtn'): Избира елемента с id aboutBtn, който представлява бутонът за показване на менюто "About".

document.getElementById('aboutMenu'): Избира елемента с id aboutMenu, който представлява самото меню "About".

document.getElementById('closeBtn'): Избира елемента с id closeBtn, който представлява бутонът за затваряне на менюто "About".

**Добавяне на слушател за събитие click на aboutBtn**

aboutBtn.addEventListener('click', ...): Добавя слушател за събитие click към бутона aboutBtn.

aboutMenu.style.display = 'flex': Когато бутонът е натиснат, стилът display на менюто aboutMenu се променя на flex, което го прави видимо на екрана.

**Добавяне на слушател за събитие click на closeBt**

closeBtn.addEventListener('click', ...): Добавя слушател за събитие click към бутона closeBtn.

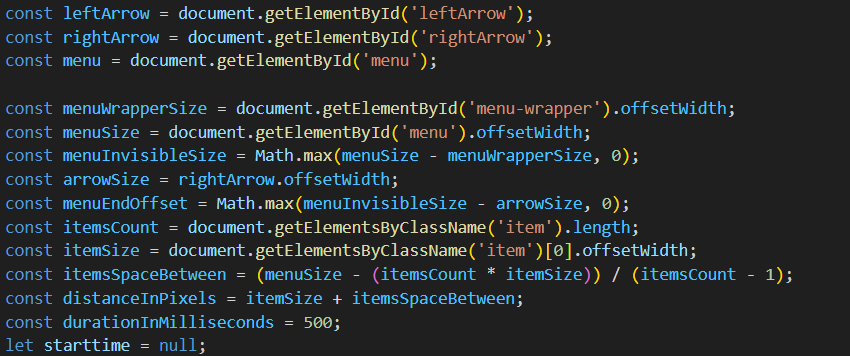
aboutMenu.style.display = 'none': Когато бутонът е натиснат, стилът display на менюто aboutMenu се променя на none, което го скрива от екрана.

**Добавяне на слушател за събитие click на прозореца (window)**

window.addEventListener('click', ...): Добавя слушател за събитие click към целия прозорец.

if (event.target === aboutMenu): Проверява дали целта на събитието (елементът, върху който е кликнато) е самото меню aboutMenu.

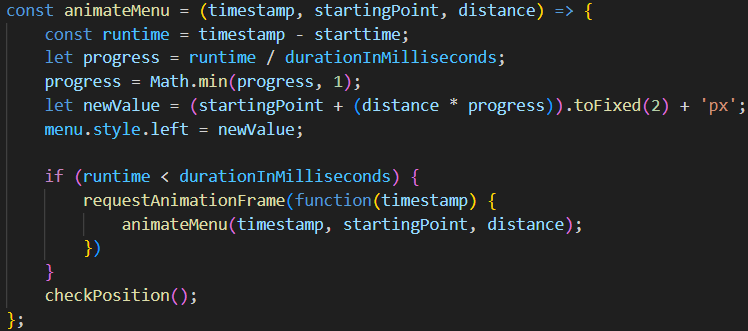
aboutMenu.style.display = 'none': Ако условието е изпълнено, менюто aboutMenu се скрива чрез промяна на стила display на none.



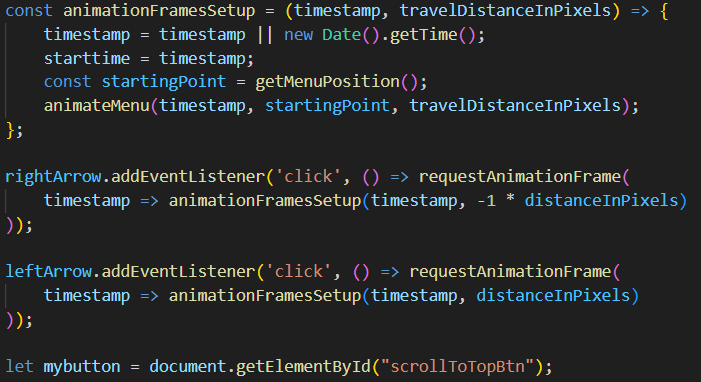
1. const leftArrow = document.getElementById('leftArrow'); и const rightArrow = document.getElementById('rightArrow');: Тези два реда взимат елементите от HTML документа, които се идентифицират с ID-тата 'leftArrow' и 'rightArrow'. Предполага се, че те са стрелките за навигация на менюто.
2. const menu = document.getElementById('menu');: Тук се взима елементът с ID 'menu', който представлява самото меню.
3. const menuWrapperSize = document.getElementById('menu-wrapper').offsetWidth;: Този ред взима ширината на контейнера на менюто.
4. const menuSize = document.getElementById('menu').offsetWidth;: Тук се взима ширината на менюто.
5. const menuInvisibleSize = Math.max(menuSize - menuWrapperSize, 0);: Пресмята се разликата между ширината на менюто и контейнера му, за да се определи колко от менюто е извън обхвата на контейнера.
6. const arrowSize = rightArrow.offsetWidth;: Взима се ширината на стрелките за навигация.
7. const menuEndOffset = Math.max(menuInvisibleSize - arrowSize, 0);: Пресмята се максималното отдалечение, което може да се превърти в менюто вдясно, за да стане невидимо.
8. const itemsCount = document.getElementsByClassName('item').length;: Брои колко елемента от клас 'item' има в менюто.
9. const itemSize = document.getElementsByClassName('item')[0].offsetWidth;: Взима се ширината на първия елемент с клас 'item' в менюто.
10. const itemsSpaceBetween = (menuSize - (itemsCount \* itemSize)) / (itemsCount - 1);: Пресмята се разстоянието между елементите в менюто, което се дели на броя на елементите минус единица.
11. const distanceInPixels = itemSize + itemsSpaceBetween;: Тук се определя колко пиксела трябва да се премести менюто, когато се навигира.
12. const durationInMilliseconds = 500;: Дължината на времето, което ще отнеме анимацията на преместване на менюто.
13. let starttime = null;: Променлива, която ще покаже кога е започнала анимацията.



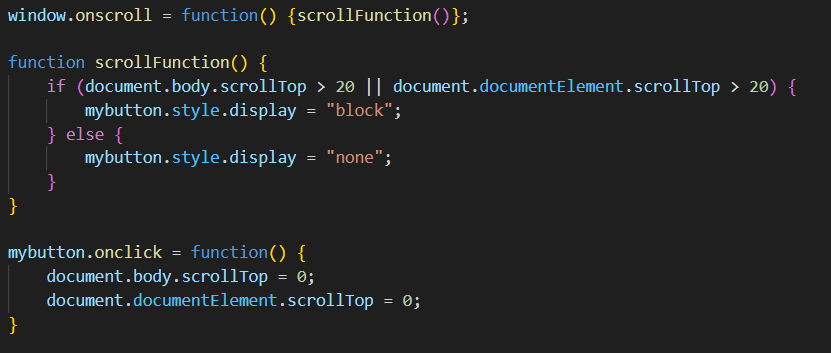
1. if (menuInvisibleSize === 0) { rightArrow.classList.add("hidden"); }: Тази проверка се изпълнява, ако менюто е напълно видимо в контейнера му. В този случай, класът "hidden" се добавя към дясната стрелка за навигация, което я прави невидима.
2. const getMenuPosition = () => { return parseFloat(menu.style.left) || 0; };: Тази функция връща текущата позиция на менюто в пиксели, като парсва стиловете на менюто за left стойността или връща 0, ако такава стойност не е зададена.
3. const getScrolledDistance = () => { return -1 \* getMenuPosition(); };: Тази функция връща отрицателната стойност на текущата позиция на менюто, представяйки колко далеч е менюто се е преместило в ляво.
4. const checkPosition = () => { const menuPosition = getScrolledDistance(); ... };: Тази функция проверява текущото положение на менюто и в зависимост от това променя видимостта на стрелките за навигация.
5. if (menuPosition <= arrowSize) { ... }: Ако менюто е преместено до края наляво, по-малко или равно на ширината на стрелките, тоест в началото на менюто, лявата стрелка става невидима, докато дясната става видима.
6. else if (menuPosition < menuEndOffset) { ... }: Ако менюто е преместено на по-голямо разстояние от края наляво отколкото може да стане невидимо, и по-близо до края отдясно, и двете стрелки са видими.
7. else if (menuPosition >= menuEndOffset) { ... }: Ако менюто е преместено до края отдясно и вече не може да се превърти невидимо в този посока, дясната стрелка става невидима, а лявата става видима.
8. document.querySelector("#print-menu-position span").textContent = menuPosition + 'px';: Тази линия актуализира текстовия елемент в HTML документа с идентификатор 'print-menu-position', като показва текущата позиция на менюто в пиксели.Top of FormBottom of Form



1. const animateMenu = (timestamp, startingPoint, distance) => { ... }: Това е дефиницията на функцията, която приема три параметъра: timestamp (времевата марка), startingPoint (началната позиция на менюто) и distance (разстоянието, което трябва да се премести менюто).
2. const runtime = timestamp - starttime;: Пресмятане на времето, което е изминало от стартирането на анимацията до текущия момент.
3. let progress = runtime / durationInMilliseconds;: Прогресът на анимацията се изчислява като отношението на изминалото време към общата продължителност на анимацията.
4. progress = Math.min(progress, 1);: Уверява се, че прогресът не надвишава 1, за да се предотврати прекомерно преместване на менюто.
5. let newValue = (startingPoint + (distance \* progress)).toFixed(2) + 'px';: Изчислява новата позиция на менюто, като умножава прогресът по разстоянието, което трябва да се премести, и добавя го към началната позиция на менюто. toFixed(2) се използва, за да се закръгли резултатът до две десетични цифри.
6. menu.style.left = newValue;: Задава новата стойност на свойството left на стиловете на менюто, което премества менюто към новата позиция.
7. if (runtime < durationInMilliseconds) { ... }: Проверява дали анимацията все още не е достигнала общото време на изпълнение. Ако не е, функцията използва requestAnimationFrame за да повика себе си отново със следващия timestamp.
8. checkPosition();: Извиква функцията checkPosition, която обновява видимостта на стрелките за навигация в менюто в зависимост от текущата позиция на менюто.



1. const animationFramesSetup = (timestamp, travelDistanceInPixels) => { ... }: Това е функция, която се използва за подготовка на анимацията. Тя приема два параметъра: timestamp (времевата марка) и travelDistanceInPixels (разстоянието, което трябва да премине менюто).
2. timestamp = timestamp || new Date().getTime();: Тук се проверява дали timestamp е инициализиран. Ако не е, се използва текущото време като стойност.
3. starttime = timestamp;: Задава се началната времева марка на анимацията.
4. const startingPoint = getMenuPosition();: Взима се текущата позиция на менюто.
5. animateMenu(timestamp, startingPoint, travelDistanceInPixels);: Стартира анимацията, извиквайки функцията animateMenu с подадените аргументи.
6. rightArrow.addEventListener('click', () => requestAnimationFrame( timestamp => animationFramesSetup(timestamp, -1 \* distanceInPixels) ));: Този код добавя event listener за клик на дясната стрелка за навигация. При клик се извиква функцията requestAnimationFrame, която стартира анимацията, като предава отрицателното разстояние, което трябва да се премине назад (наляво).
7. leftArrow.addEventListener('click', () => requestAnimationFrame( timestamp => animationFramesSetup(timestamp, distanceInPixels) ));: Тук се добавя event listener за клик на лявата стрелка за навигация. При клик се извиква функцията requestAnimationFrame, която стартира анимацията, като предава положителното разстояние, което трябва да се премине напред (надясно).
8. let mybutton = document.getElementById("scrollToTopBtn");: Това изглежда като начало на селекция на HTML елемент с id "scrollToTopBtn", но липсва присвояването на стойност или логика, свързана с този елемент. Вероятно този ред от код е непълно представен или част от по-голяма част от кода, която не е предоставена.



1. window.onscroll = function() {scrollFunction()};: Това установява event listener, който се активира при скролиране на прозореца. Когато се скролира, се извиква функцията scrollFunction().
2. function scrollFunction() { ... }: Тази функция се извиква при скролиране и проверява дали превъртането на страницата е достатъчно голямо (повече от 20 пиксела). Ако е така, бутона за превъртане до началото на страницата (mybutton) се показва (display стилът става "block"), в противен случай той се скрива (display стилът става "none").
3. mybutton.onclick = function() { ... }: Тази част от кода добавя event listener към бутона за превъртане до началото на страницата. Когато бутона се кликне, се изпълнява функцията, която задава scrollTop свойствата на document.body и document.documentElement на 0. Това премества изгледа на страницата към началото.

**Четвърта глава:**

Във нашия уебсайт първо ще забележите информация за болида Mcl60. Тази секция предоставя обща представа за болида, включително кога той дебютира във Формула 1 и какви постижения е постигнал през своя сезон в състезанието.

Следващото, което ще намерите в нашия сайт, е впечатляващ 3D модел на болида. В тази част на уебсайта са отбелязани най-важните части от аеродинамиката и начина на работа на болида, представени с изключителна прецизност и детайли.

Продължавайки с разглеждането, ще откриете подробна техническа информация за двигателя, аеродинамиката, гумите и спирачките на болида. Тази секция предлага дълбоко разучена аналитика за всеки аспект на технологията зад Mcl60.

Накрая, нашият уебсайт завършва с впечатляващо видео на представянето на болида и галерия от снимки. Това позволява на посетителите да се потопят във визуалното вълшебство на Mcl60, докато гледат как той преминава през различни аспекти на неговата експлоатация и представяне на пистата.