Створення сайту онлайн-книгарні

ЗМІСТ

3.1.2 Перенесення дизайну в HTML та CSS.

3.1.3 Верстка Header

3.1.4 Верстка Footer

3.1.5 Верстка головної сторінки

3.1.6 Верстка сторінки «Каталог»

3.1.7 Верстка сторінки «Жанри»

3.1.8 Верстка сторінки «Про книгу»

3.1.9 Створення сторінки «Читати книгу»

3.1.10 Верстка сторінки «Підтримка»

3.1.3 Додавання інтерактивності за допомогою JavaScript.

3.2 Перевірка роботи сайту на різних пристроях і браузерах.

3.3 Виявлення і виправлення помилок у відображенні та функціональності.

3.4 Покращення швидкодії сайту.

3.5 Оптимізація для SEO.

3.6 Вдосконалення адаптивності до різних пристроїв і роздільних здатностей екрану.

РОЗДІЛ 4

Висновки та додатки

4.1 Підбиття підсумків процесу розробки.

4.2 Аналіз отриманих результатів та висновків.

4.3 Висунення рекомендацій для подальших покращень.

4.4 Додатки

4.4.1 Код HTML, CSS, JS.

4.4.2 Знімки екрану з макету в Figma.

4.4.3 Звіт про тестування.

4.4.4 Посилання на готовий проект.

Вступ

Опис мети курсової роботи.

Актуальність теми

У сучасному світі веб-сайти та веб-додатки стали невід'ємною частиною інформаційних технологій. Через те, що Інтернет є основним джерелом інформації для більшості людей, веб-сайти та веб-додатки набули великої популярності. Веб-сайти дозволяють компаніям, установам та індивідуумам стати доступними своїм клієнтам та аудиторії 24/7.

Раніше сайти мали обмежений доступ та можливості, але зараз надають найсучасніші та найрізноманітніші послуги: електронну комерцію, комунікації та взаємодії, бізнес-послуги тощо.

Справжнім проривом стало те, що сайти стали доступні на смартфонах та різних пристроях завдяки адаптації під різні пристрої.

Зараз важко переоцінити розвиток веб-технології, вони постійно розвиваються та мають постійне застосування серед користувачів. У міру розвитку самих технологій зростає доступність та зручність користування веб-сайтами та веб-додатками.

Пояснення вибору інструментів (HTML/CSS/JS для реалізації).

Для створення сайту було обрано так загальновідомі інструменти, як HTML, CSS та JavaScript.

HTML – це мова розмітки, яка використовується для розміщення основної інформації на сайті – фото, тексту, відео тощо.

SCSS

Розділ 1

Постановка задачі

1.1 Теоретичний огляд матеріалів та макету

Для створення сайту ми використаємо готовий макет для веб-сайту онлайн-книгарні. Макет створений та доступний у Figma. На рис. 1 видно цей макету сайту.

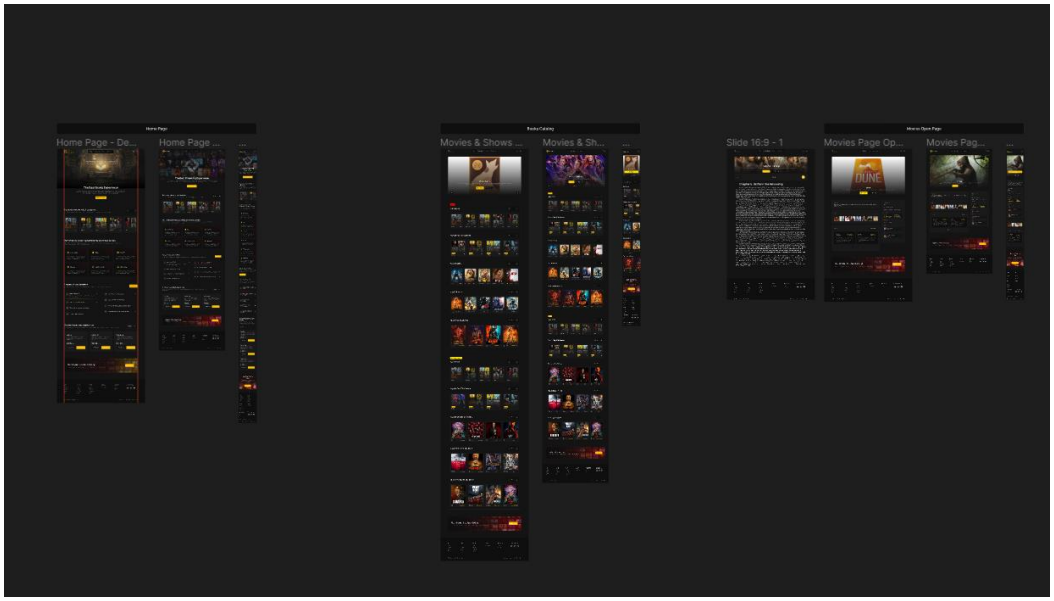


Рис. 1. Макет сайту в Figma

Даний макет складається із кількох сторінок із різним вмістом.

1. Головна сторінка. Тут розміщено ознайомчу інформацію про вид діяльності сайту, його послуги та навігаційна панель.

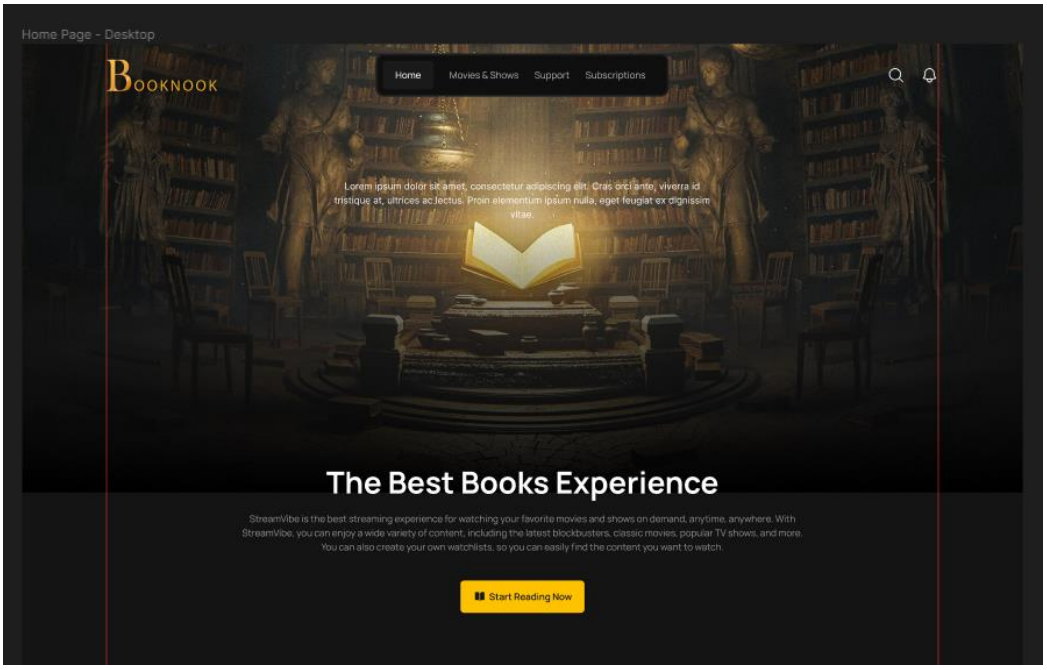


Рис. 2. Частина макету головної сторінки

2. Каталог. Тут розміщенні усі види та жанри книг, наукової літератури тощо, які користувач може обрати для себе. Це є одна із найголовніших сторінок, адже включає в себе навігацію між наявними книгами та документами. Тобто кожного жанру книг буде своя власна сторінка, яка генеруватиметься із наявних книг цього жанру. Далі ми можемо перейти до кожної книги, подивитись основну інформацію про неї і за бажанням прочитати.
3. Сторінка підтримки. Тут можна буде залишити власний відгук або зв’язатися із адміністрацією сайту, якщо у користувачів виникнути проблеми чи запитання.

1.2 Визначення функціональних вимог до сайту.

Найголовніша вимога до сайту – надавати якісні послуги, чудово взаємодіяти із користувачем, реагувати на його дії тощо.

Правильний веб-сайт має мати ряд вимог:

Доступність

Адаптивність

Швидкість

Семантичність

Простота

Взаємодію тощо.

Розділ 2
Аналіз інструментів за засобів для створення сайтів

2.1 Середовище розробки Visual Studio Code

Visual Studio Code — це дуже зручний засіб для створення, редагування та зневадження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X. Компанія Microsoft представила Visual Studio Code у квітні 2015 на конференції Build 2015.

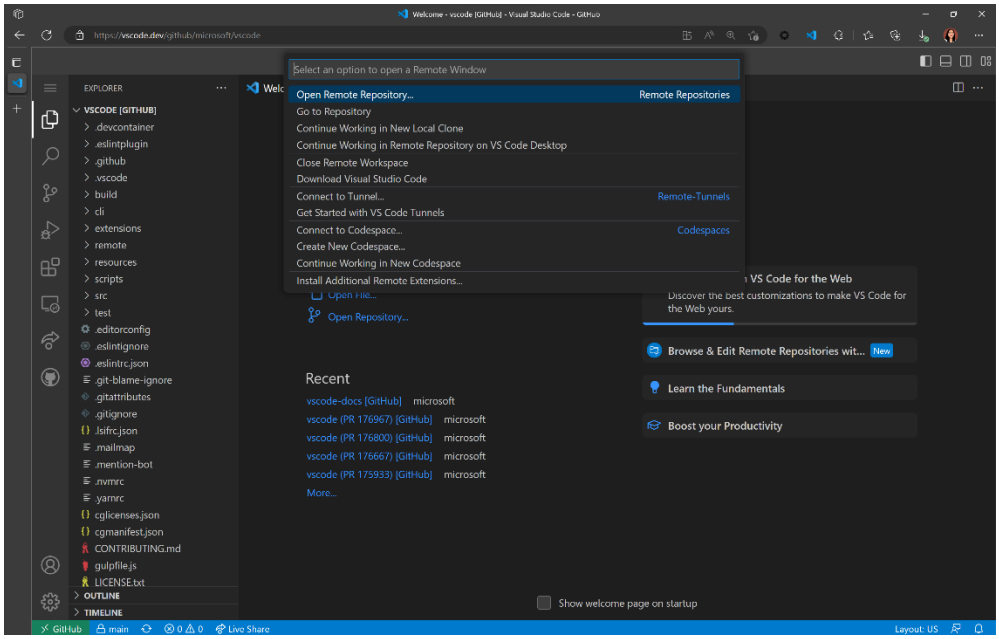


Рис. . Середовище розробки Visual Studio Code

Visual Studio Code дуже зручний засіб для написання коду та має ряд переваг. Замість системи проєктів вона дозволяє користувачам відкривати один або кілька каталогів, які потім можна зберегти в робочих областях для подальшого використання. Це дозволяє йому працювати як мовно-агностичний редактор коду для будь-якої мови. Він підтримує багато мов програмування та набір функцій, який відрізняється для кожної мови. Небажані файли та папки можна виключити з дерева проєкту за допомогою налаштувань. Багато функцій Visual Studio Code не доступні через меню чи інтерфейс користувача, але доступ до них можна отримати через палітру команд.

Код Visual Studio можна розширити за допомогою розширень, доступних через центральне сховище. Це включає доповнення до редактора та підтримку мови. Примітною функцією є можливість створювати розширення, які додають підтримку нових мов, тем, налагоджувачів, налагоджувачів подорожей у часі, виконують статичний аналіз коду та додають лінтери коду за допомогою протоколу Language Server.

Керування джерелом є вбудованою функцією Visual Studio Code. Він має спеціальну вкладку всередині панелі меню, де користувачі можуть отримати доступ до налаштувань керування версіями та переглянути зміни, внесені до поточного проєкту. Щоб використовувати цю функцію, Visual Studio Code має бути зв'язано з будь-якою підтримуваною системою керування версіями (Git, Apache Subversion, Perforce тощо). Це дозволяє користувачам створювати репозиторії, а також робити запити push і pull безпосередньо з програми Visual Studio Code.

Visual Studio Code містить кілька розширень для FTP, що дозволяє використовувати програмне забезпечення як безплатну альтернативу для веб-розробки. Код можна синхронізувати між редактором і сервером без завантаження додаткового програмного забезпечення.

Код Visual Studio дозволяє користувачам встановлювати кодову сторінку, на якій буде збережено активний документ, символ нового рядка та мову програмування активного документа. Це дозволяє використовувати його на будь-якій платформі, у будь-якій локальній мережі та для будь-якої заданої мови програмування.

Visual Studio Code збирає дані про використання та надсилає їх до Microsoft, хоча це можна вимкнути. Через відкритий вихідний код програми, код телеметрії доступний для громадськості, яка може бачити, що саме збирається.

2.2 Інструменти розробки: мова розмітки HTML, стилі CSS, мова програмування JavaScript

Для створення веб-сайту зазвичай використовують три основні інструменти:

- Мова розмітки HTML

- Каскадні таблиці стилів CSS
- Мова програмування JavaScript

HTML відіграє важливу роль у веб-розробці, оскільки визначає структуру та вміст веб-сторінок. Він служить основою, на якій ґрунтуються веб-сайти. HTML досягає цього, використовуючи систему тегів і елементів, кожен з яких має унікальне призначення для додавання інформації та контенту на сторінки.

CSS - це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних (HTML/XML). CSS є основною технологією всесвітньої павутини, поряд із HTML та JavaScript. Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат **CSS** може застосовуватися до інших видів XML-документів. CSS (каскадне або блокове верстання) прийшло на заміну табличному верстанню веб-сторінок. Головна перевага блокового верстання — розділення вмісту сторінки (даних) та його візуальної презентації.

JavaScript (**JS**) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Отже, для створення веб-сайту книги будуть застосовуватися усі ці три популярні інструменти.

2.3 Методологія БЕМ

Окремо для HTML буде використовуватись спеціальна методологія для написання класів та розмітки – методологія БЕМ.

Методологія БЕМ — одна з найпопулярніших на даний момент, в основі якої лежить домовленість про те, як розбивати інтерфейс на певні незалежні блоки. Поняття та суть методології полягає у розбитті сайту на різні частини, які представленні в її абревіатурі як «Блок Елемент Модифікатор». Цей підхід до веб-розробки дозволяє швидко створювати сайти з гнучкою архітектурою.

БЕМ має ряд вимог та правил для створення розмітки. Деякі з них наведені нижче:

БЕМ-блок – незалежний самодостатній об’єкт, що не впливає на оточення (сусідні елементи). Тобто при його видаленні він не впливає на структуру сторінки, його можна використовувати на інших сторінках

БЕМ-елемент – залежний об’єкт від БЕМ-блоку. Завжди знаходить у БЕМ-блокові та вкладений у нього, може впливати на оточення (сусідні елементи).

БЕМ-модифікатор – дозволяє вносити певні зміни для БЕМ-блоків та БЕМ-елементів.

БЕМ-мікс – одночасне використання БЕМ-блоку та БЕМ-елементу для одного об’єкту.

Детальніше цей підхід буде показано на практиці.

2.4 Препроцесор SASS/SCSS. Попередні налаштування проекту

SASS - це препроцесор CSS, який допомагає зменшити повторення з CSS і, в свою чергу, економить час. Він вважається більш стабільним і потужним в порівнянні з CSS. Він описує стиль документа дуже чітко та структурно. Це допомагає швидше і якісніше виконувати роботу. При всіх своїх особливостях він переважно віддається перевазі над CSS. Це дозволяє визначати змінні, які можуть починатися зі знаку $. Призначення змінної можна виконати за допомогою двокрапки. Він також підтримує логічне вкладення, яке не має CSS. SASS дозволяє користувачеві вкладений код, який можна вставити один у одного.

SCSS – синтаксис SASS, яким ми будемо використовувати на практиці. Його перевага полягає у більшій зручності порівняно із CSS: вкладеність, змінні, міксіни тощо.

Розділ 3

Програмна реалізація.

Будь яка програмна реалізація потребує підготовки та налаштування. Як говорилося раніше, для розробки сайт буде використовуватися середовище розробки Visual Studio Code, а також такі інструменти для реалізації як HTML/CSS/JS і препроцесор SASS.

3.1 Розробка фронтенду

3.1.1 Розбиття макету на окремі компоненти та сторінки.

Першим та дуже важливим розділом на початку створення сайту є аналіз макету. Як говорилося раніше, наш макет – це сайт, який надає послуги оцінок та рецензій на популярні книги, розповідає інформацію про них та іноді дозволяє читати певні книги. Також на сайті буде розміщено наукову літературу, яка публікуватиметься університетом.

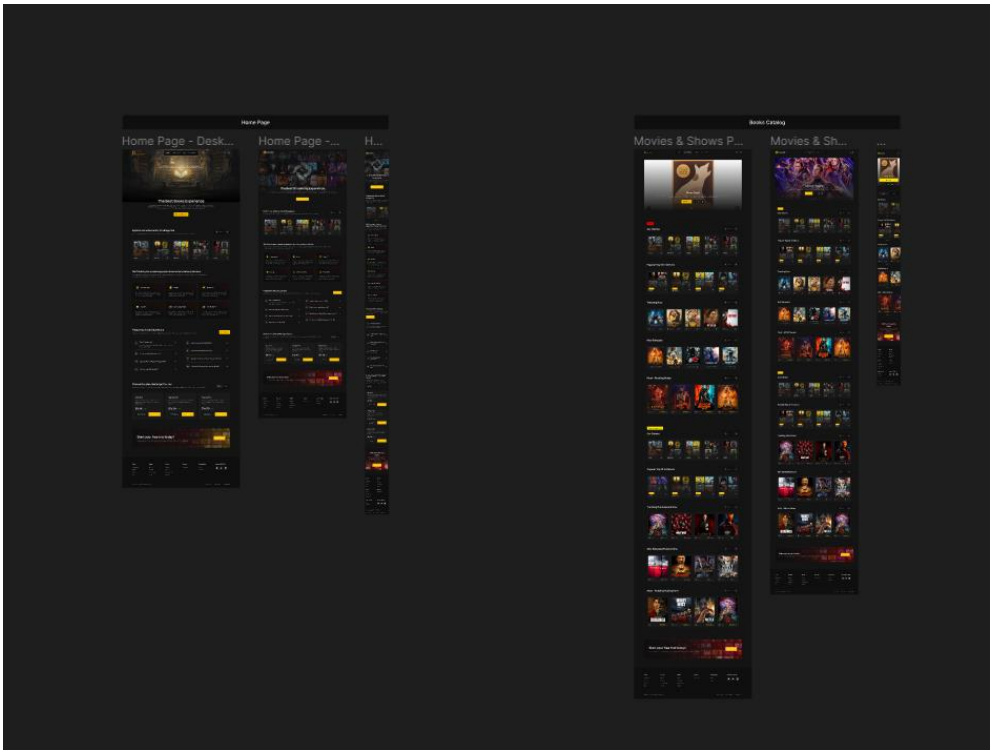


Рис. . Фото макету сайту

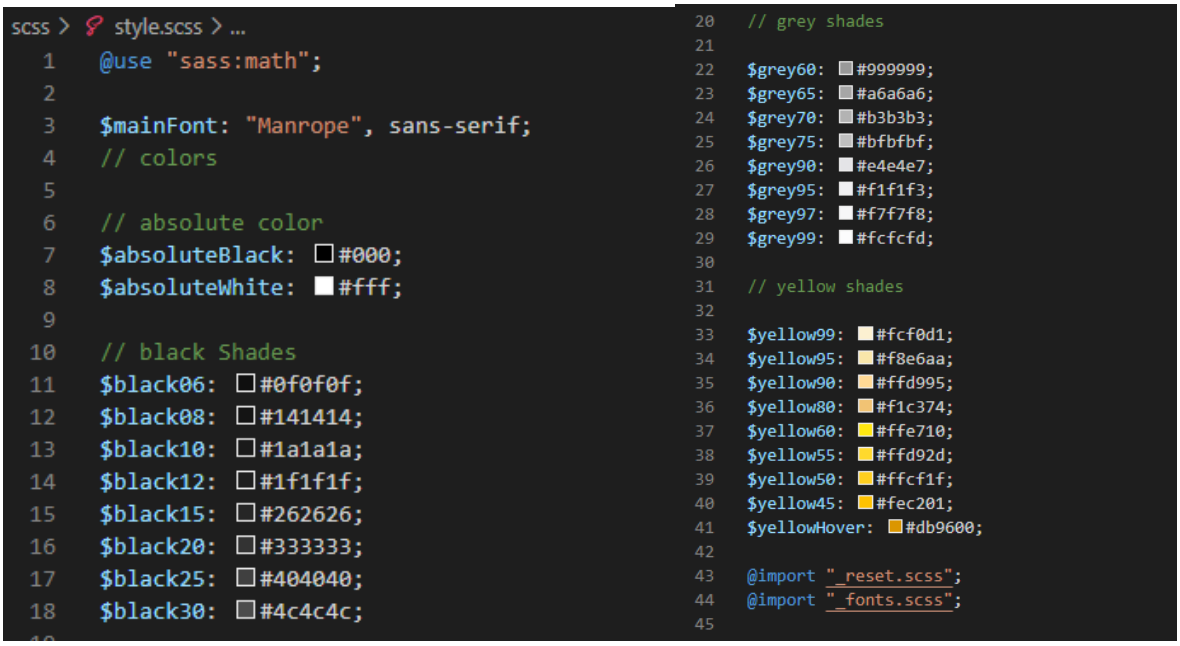
3.1.2 Перенесення дизайну в HTML та CSS.

Найпершим та найголовнішим кроком, коли починається процес роботи із HTML/CSS є визначення усіх шрифтів, кольорів та іконок, які мають бути на сайті.

Спочатку потрібно підключити ті шрифти, які дизайнер використовував в макеті, їх жирність та стиль тощо. Потім потрібно додати ці стилі в окремий файл SCSS (_fonts.scss).

Завдяки можливостям SCSS ми можемо створити змінні із кольорами, які використовуються на сайті.

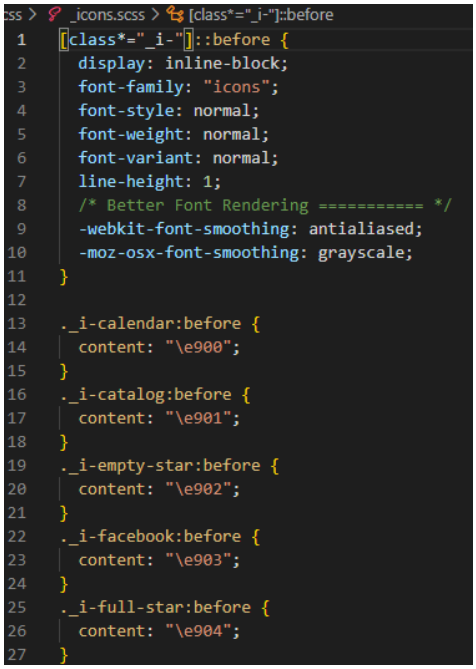
Також необхідно додати файл, який міститиме скидання стилів (_reset.scss), - правила, які забирають стилі створенні браузером за замовчуванням.



```scss
@use "sass:math";

$mainFont: "Manrope", sans-serif;
// colors

// absolute color
$absoluteBlack: #000;
$absoluteWhite: #fff;

// black Shades
$black06: #0f0f0f;
$black08: #141414;
$black10: #1a1a1a;
$black12: #1f1f1f;
$black15: #262626;
$black20: #333333;
$black25: #404040;
$black30: #4c4c4c;

// grey shades

$grey60: #999999;
$grey65: #a6a6a6;
$grey70: #b3b3b3;
$grey75: #bfbfbf;
$grey90: #e4e4e7;
$grey95: #f1f1f3;
$grey97: #f7f7f8;
$grey99: #fcfcfd;

// yellow shades

$yellow99: #fcf0d1;
$yellow95: #f8e6aa;
$yellow90: #ffd995;
$yellow80: #f1c374;
$yellow60: #ffe710;
$yellow55: #ffd92d;
$yellow50: #ffcf1f;
$yellow45: #fec201;
$yellowHover: #db9600;

@import "_reset.scss";
@import "_fonts.scss";
```

Рис. Додавання змінних SCSS та скидання правил

Також важливим кроком буде імпорт та перетворення на шрифт усіх іконок, які використовуються на сайті. Це необхідно для кращої гнучкості сайту і це можна зробити завдяки веб-ресурсу Icomoon.



```scss
[class*="_i-"]::before {
  display: inline-block;
  font-family: "icons";
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  line-height: 1;
  /* Better Font Rendering =========== */
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

._i-calendar:before {
  content: "\e900";
}
._i-catalog:before {
  content: "\e901";
}
._i-empty-star:before {
  content: "\e902";
}
._i-facebook:before {
  content: "\e903";
}
._i-full-star:before {
  content: "\e904";
}
```

Рис. Додавання іконочного шрифту

Наступним важливим кроком є виділення тих частин макету сайту, які часто та неодноразово повторюються, або є дуже подібними. Це потрібно виділити під час аналізу макету.

Для прикладу це можуть бути слайдери із зображеннями та жанрами книг. Саме такого типу слайдери будуть дуже часто зустрічатися на кількох сторінках сайту.

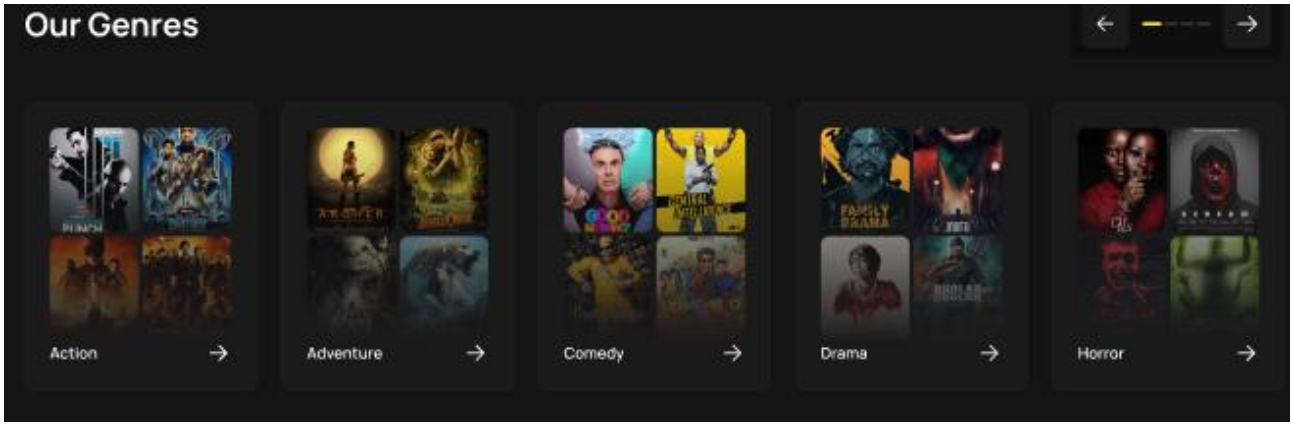


Рис. Слайдер, який неодноразово повторюється на сайті

Тому найперше, що потрібно зробити це створити окрему сторінку та файл стилів, де будуть зберігати ці об’єкти. Після закінчення роботи над сайтом їх можна видалити.

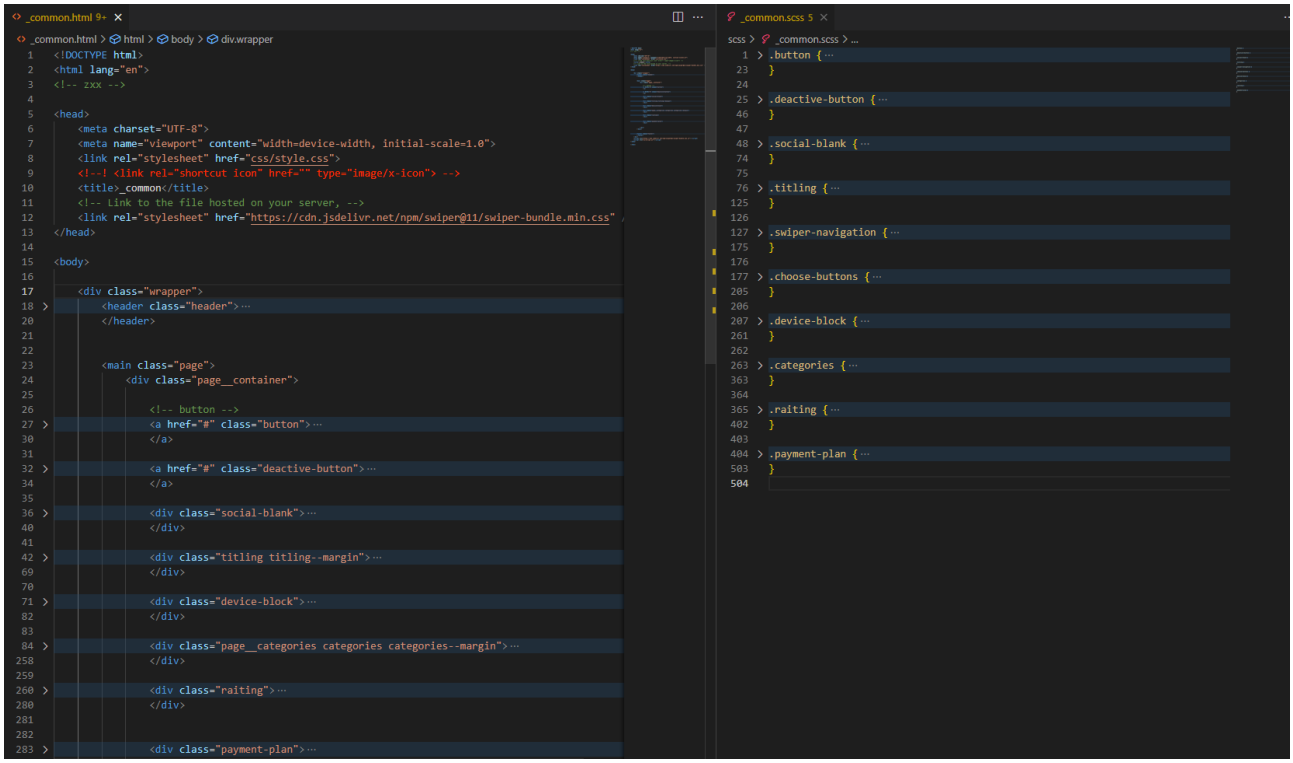


Рис. Створення окремої сторінки та файлу стилів

Після цього у нас будуть готові, об’єкти, які ми потім легко та неодноразово будемо використовувати.

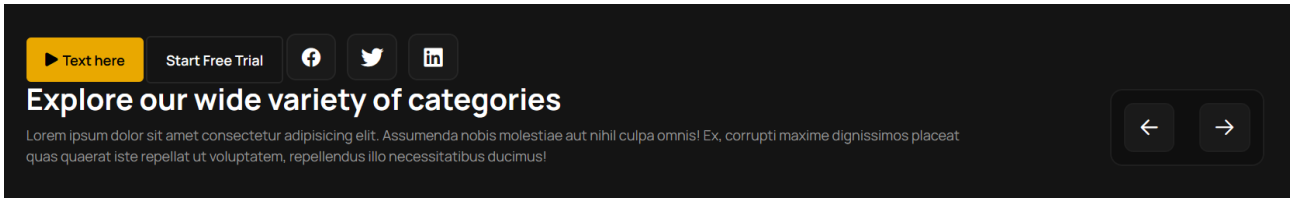


Рис. . Готовий створений елемент, який часто повторюється

3.1.3 Верстка Header

Header (шапка) – це верхня частина сайту, де зазвичай розміщений логотип, навігаційне меню та допоміжні функції. В макеті вона має такий вигляд:

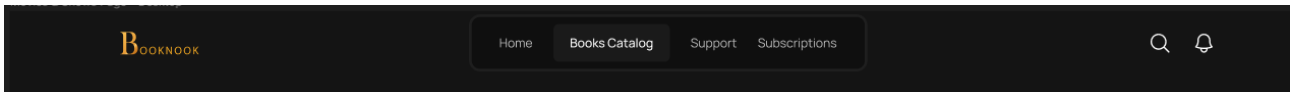


Рис. . Header

Щоб додати основну інформацію на сайт треба створити семантичний тег header, а в ньому створити обмежуючий контейнер.

Обмежуючий контейнер – це контейнер, який має обмеження по ширині, коли сайт має великий розмір.

В обмежуючий контейнер ми додаємо основні теги:

- Для логотипу використаємо тег img (на головній сторінці це буде тег фото, а на інших посилання для переходу на головну).
- Для навігації використаємо семантичний тег nav та список ul li.
- Для інших елементів, як додаткові дії, ми використаємо теги button чи a.

Рис. HTML-розмітка header

Варто звернути увагу на спосіб назви класів. Тут наочно використовується методологія БЕМ. Тег «header» є БЕМ-блоком, тому має назву класу «header». Так як БЕМ-блок не може впливати на інші елементи, ми задаємо йому БЕМ-модифікатор із назвою, що означає, що шапка буде фіксуватись на сторінці.

Усі інші елементи: «header__navigation», «header__action», «header__logo» тощо – це БЕМ-елементи. Вони напряму залежать від свого БЕМ-блоку, впливають на оточення тощо.

Яскравий приклад БЕМ-міксу є навігаційне меню, в класі якого вказано «header__menu» та «menu». Це означає, що елемент є водночас БЕМ-блоком та БЕМ-елементом. Це дозволяє вивільнити новий простір імен для класів, спростити читабельність та зробити верстку більш зручною.

Тепер потрібно звернути увагу на стилі. Для розташування усіх елементів класу «header» потрібно застосувати фіксоване позиціонування position: fixed та розмістити блок зверху сторінки. Для того щоб елементи розташовувалися горизонтально потрібно використати глобальний параметр відображення display: flex із урахуванням того, що пізніше якогось елемента може не бути. Flexдозволяє гнучко реагувати на ці зміни.

```scss
.header {
    transition: background-color 0.3s ease;
    &.scrolled {
        background-color: $black06;
    }

    &::before {
        content: "";
        width: 100%;
        height: 100%;
        position: absolute;
        left: -100%;
        top: 0;
        transition: all 0.3s;
        .menu-open & {
            z-index: 3;
            left: 0;
            background-color: $black08;
        }
    }

    &--fixed {
        position: fixed;
        left: 0;
        top: 0;
        width: 100%;
        z-index: 90;
    }

    &__container {
        display: flex;
        gap: toRem(30);
        justify-content: space-between;
        align-items: center;
    }

```

Рис. . Стилі для header

До кожного окремого елемента задаються окремі стилі. Для логотипа вказуються правила ширини та розміру зображення, щоб при заміні його на інший, не постраждала структура шапку. Для цього треба використати правила max-width, aspect-ratio та object-fit cover.

Для основного навігаційного меню можна теж використати flexbox, для більшої гнучкості.

В результаті верстки ми отримаємо готову шапку сайту.

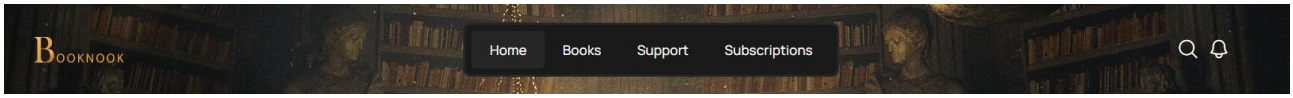


Рис. . Готова шапка сайту

Структура сторінки не страждатиме від того, що контент може різко змінитися. Але це ще не кінцевий результат. Важливою подальшою складовою є адаптація цього блоку під різні пристрої, особливо смартфони. Видно, що на певній ширині екрану навігаційне меню не влазить на головному екрані, тому для цього створюється випадаюче меню.

Для цього потрібно використати медіа-запити та JS. Можна припустити, що найкраща точка, коли потрібно змінювати вигляд шапки – це ширина екрану в 820 пікселів.

Тому завдяки властивості @media при цій ширину ми змінюємо зовнішній вигляд. На рисунку видно, що ми змінюємо декілька правил навігаційного меню та приховуємо його.

```
&__navigation {
    position: relative;
    z-index: 4;
    @media (max-width: toEm(820)) {
        overflow: auto;
        position: fixed;
        z-index: 1;
        background-color: $yellow55;
        padding: toRem(130) toRem(20) toRem(40);
        left: -100%;
        top: 0;
        height: 100%;
        width: 100%;
        display: flex;
        gap: toRem(40);
        flex-direction: column;
        //align-items: flex-end;
        transition: all 0.3s;
        .menu-open & {
            left: 0;
        }
    }
}
```

Рис. . Медіа запит для навігаційного меню

Також при цій самій ширині ми показуємо кнопку меню-бургер (display: block із none) та відповідно стилізуємо її.

Отже, тепер шапка сайту має такий вигляд:

Рис. . Вигляд шапки сайту після використання медіа запиту

Наступним кроком стане реалізувати відкриття меню при натисканні на меню-бургер. Це можна зробити завдяки JS.

Ми створюємо подію-прослуховувач addEventListener та задаємо їй функцію, яка виконається при клікові. Потім ми реалізовуємо цю функцію. При клікові ми визначаємо на який елемент ми натиснули та присвоюємо це значення константі.

Потім ми створюємо умову, де перевіряємо чи натиснутий елемент має клас меню-бургера. Для цього використовуємо closest, а не classList.constains() через одну перевагу цього методу (closest спрацює навіть, якщо ми натиснемо на дочірній елемент кнопки, наприклад span).

Коли умова спрацьовує, ми додаємо до тегу body в HTML додатковий клас “menu-open”. classLIst.toggle дозволяє додавати клас, а при повторному клікові прибирати його, що дуже зручно.

```
document.addEventListener("click", documentActions)

function documentActions(e) {
    const targetElement = e.target
    if (targetElement.closest(".header__burger-block")) {
        document.body.classList.toggle("menu-open")
    }
}
```

Рис. . Додавання додаткових класів через JS

Потім ми прописуємо в SCSS, що якщо body має клас “menu-open”, то показуємо меню користувачеві, додавши анімацію. Результат на рисунку.

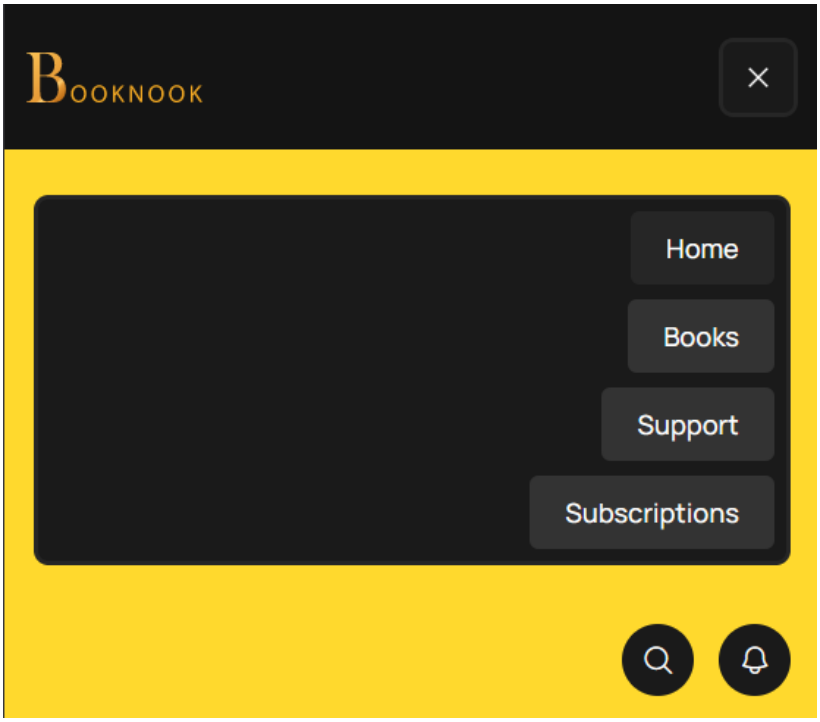


Рис. . Показ меню при клікові

Також ми завдяки медіа запитам ми можемо змінювати інші правила: зменшити розмір тексту, колір та розміри для кращого відображення на менших пристроях. Після цього можна зробити висновок, що шапка сайту повністю зверстана та та може використовуватись на інших сторінках.

3.1.4 Верстка Footer

Footer має роль додаткової блоку, який містить менш важливу інформацію та навігацію, а також зазвичай містить політику конфіденційності та посилання на соціальні мережі.

В макеті теж видно, що макет футера має усі ці критерії і з них усіх можна виділити дві окремі частини.

У першій буде розташовані схожі списки навігації, що можна зробити завдяки тегам ul/li та правилові display: flex, щоб розтягнути їх по ширині. Посилання на соціальні мережі це теж той самий блок, проте в ньому лише змінений внутрішній контент.

Другою частиною є марка та посилання на політику конфіденційності, права використання тощо.

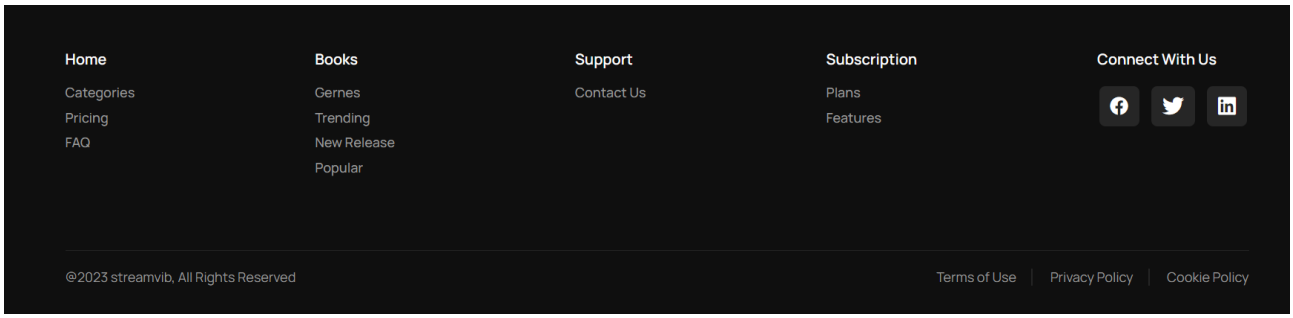


Рис. . Макет підвалу сайту

На рисунку видно HTML-розмітку для footer. Вона містить усі ті критерії, перелічені вище із використанням БЕМ-методології.

```html
<footer class="footer">
  <div class="footer__container">
    <div class="footer__body">
      <article class="footer__block-menu footer-menu-block">
        <h4 class="footer-menu-block__title">Home</h4>
        <ul class="footer-menu-block__list">
          <li class="footer-menu-block__item">
            <a href="#" class="footer-menu-block__link">Categories</a>
          </li>
          <li class="footer-menu-block__item">
            <a href="#" class="footer-menu-block__link">Pricing</a>
          </li>
          <li class="footer-menu-block__item">
            <a href="#" class="footer-menu-block__link">FAQ</a>
          </li>
        </ul>
      </article>
      <article class="footer__block-menu footer-menu-block">···
      </article>
      <article class="footer__block-menu footer-menu-block">···
      </article>
      <article class="footer__block-menu footer-menu-block">···
      </article>
      <article class="footer__block-menu footer-menu-block">···
      </article>
    </div>
    <div class="footer__bottom bottom-footer">
      <div class="bottom-footer__sign">@2023 streamvib, All Rights Reserved</div>
      <div class="bottom-footer__other">
        <a href="#">Terms of Use</a>
        <a href="#">Privacy Policy</a>
        <a href="#">Cookie Policy</a>
      </div>
    </div>
  </div>
</footer>
```

Рис. . HTML-розмітка підвалу сайту

У «footer__body» будуть вставлені блоки із навігацією. Навігація тег будуватиметься завдяки flex, проте із зміненою віссю на вертикаль. Це потрібно для гнучкості, якщо адміністрація сайту захоче прибрати один елемент із списку. Самі блоки теж робимо flex-елементами і дозволяємо їм переноситись автоматично при зменшенні екрану завдяки властивості flex-wrap: wrap;

```scss
  &__body {
    width: 100%;

    display: flex;
    align-items: flex-start;
    flex-wrap: wrap;
    justify-content: space-between;

    column-gap: toRem(55);
    row-gap: toRem(30);

    @include adaptiveValue("padding-bottom", 100, 50);
    border-bottom: toRem(1) solid $black15;

    @media (max-width: toEm(500)) {
      justify-content: space-around;
    }
  }

}
.footer-menu-block {
  display: flex;
  gap: toEm(24, 18);
  flex-direction: column;
  align-items: flex-start;
```

Рис. . Приклад стилів для блоків підвалу сайту

Іншу частину підвалу, де міститься приватний політ та правила конфіденційності можна розташувати завдяки flexbox та автоматичному переносі flex-wrap.
Тоді підвал матиме такий вигляд:

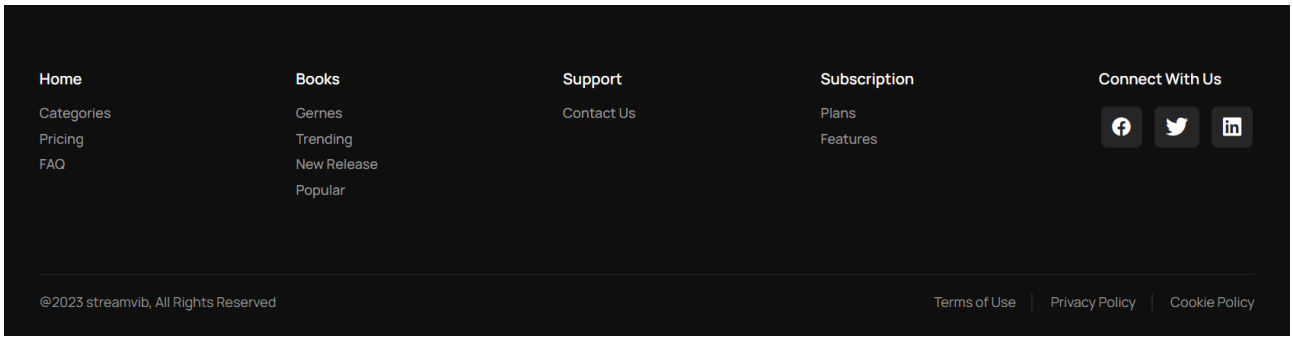


Рис. . Готова верстка підвалу для широких екранів

Адаптація під різні пристрої буде відбуватися завдяки властивості flex-wrap, а нам доведеться змінити лише розмір шрифту та відступів на певній ширині.
Тоді ми отримаємо повність адаптований підвал сайту:

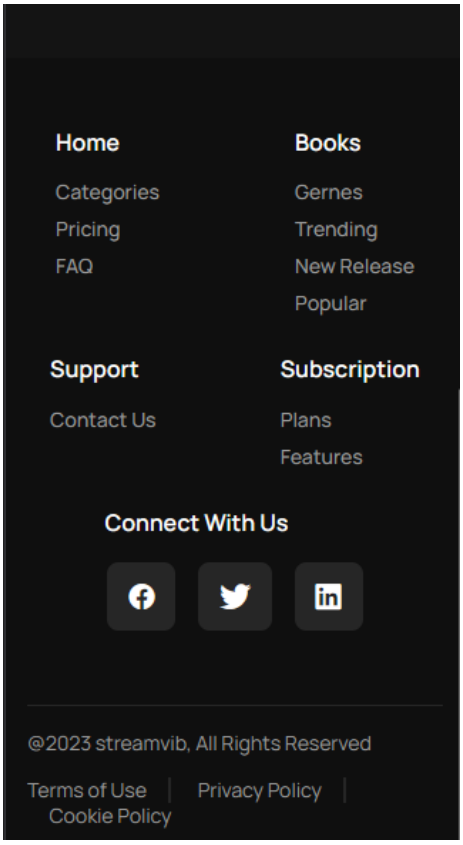


Рис. . Вигляд підвалу при ширині екрану в 322 пікселя

Через те що footer та header однакові на усіх сторінках сайту, тому вони тепер використовуватимуться, як шаблони на інших сторінках і більше не потребують верстки.

3.1.5 Верстка головної сторінки

Головна сторінка містить основну інформацію про сайт і має такі блоки, як hero, слайдер із жанрами книг, відповіді на часті запитання тощо.
Почнемо із секції hero. Це буде фонове зображення в ширину всієї сторінки, де буде знаходитись цитата про книги. Нижче буде заголовок, текст-опис та кнопка-посилання.

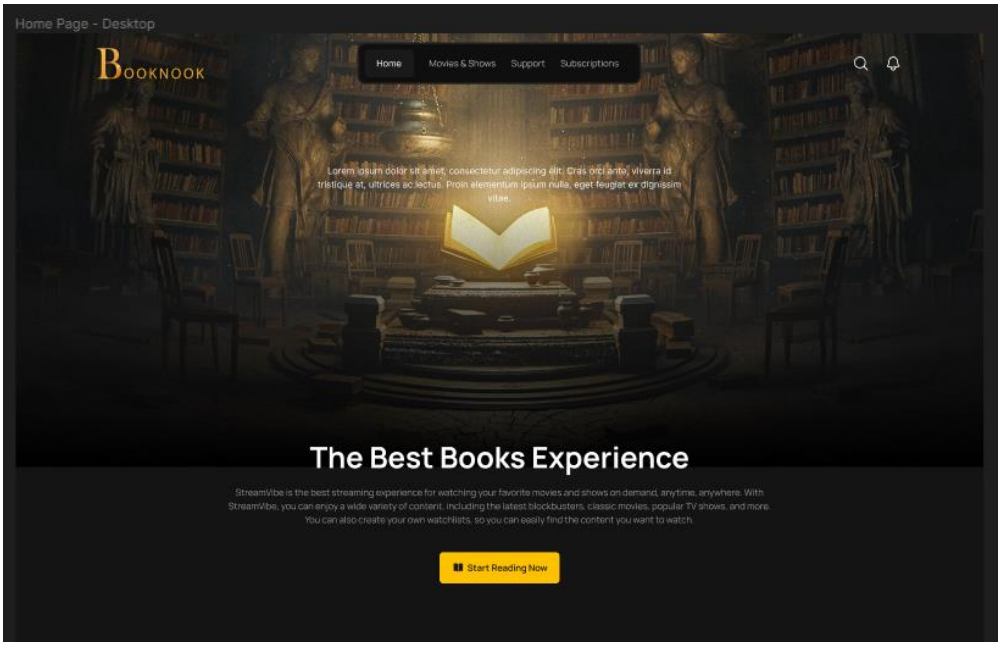


Рис. . Секція Hero в макеті

Для реалізації завдання треба зробити HTML-розмітку:

```html
<section class="page__hero hero hero--margin">
    <div class="hero__background">
        <img src="img/page/hero/kniga.jpg" alt="back-image" class="hero__image">
        <div class="hero__text-decoration">
            Lorem ipsum dolor sit adolore asperiores quas
            dolorum magni sit omnis!
        </div>
    </div>
    <div class="hero__container">
        <div class="hero__body">
            <div class="hero__content">
                <h1 class="hero__title">The Best Book Experience</h1>
                <div class="hero__text">
                    <p>
                        Lorem ipsum dolor sit adolore asperiores quas
                        dolorum magni sit omnis! Lorem ipsum dolor sit adolore asperiores quas
                        dolorum magni sit omnis! Lorem ipsum dolor sit adolore asperiores quas
                        dolorum magni sit omnis! Lorem ipsum dolor sit adolore asperiores quas
                        dolorum magni sit omnis! Lorem ipsum dolor sit adolore asperiores quas
                        dolorum magni sit omnis! Lorem ipsum dolor sit adolore asperiores quas
                        dolorum magni sit omnis!
                    </p>
                </div>
                <a href="book-catalog.html" class="button">
                    <span class="_i-read"></span>    BEM - Class names must be in kebab case
                    Start Now
                </a>
            </div>
        </div>
    </div>
</section>
```

Рис. . HTML-розмітка для секції Hero

```scss
.hero {
    &--margin {
        @include adaptiveValue("margin-bottom", 200, 100);
    }

    &__background {
        position: relative;
        &::before {
            content: "";
            position: absolute;
            width: 100%;
            height: 100%;
            left: 0;
            top: 0;
            background: url("../img/page/hero/shade.png") 0 0 / cover no-repeat;
            z-index: 1;
        }
    }

    &__image {
        position: relative;
        width: 100%;
        height: 100%;
        aspect-ratio: 1921 / 860;
        object-fit: cover;
        @include adaptiveValue("min-height", 860, 500);
    }

    &__decoration {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        aspect-ratio: 1 / 1;
        object-fit: cover;
        width: 100%;
        @include adaptiveValue("max-width", 200, 100);
    }
```

Рис. . Приклад деяких стилів для секції Hero

Ми задаємо такі основні як колір, ширину та розміри шрифту. Блок із текстом потрібно посунути вгору, щоб накласти на фото. Це можна зробити завдяки від’ємному значенню margin-top. Для того, щоб фото було адаптивним задаємо йому object-fit: cover, 100% ширини та фіксовану висоту, яку будемо потім завдяки медіа запитам зменшувати. Тоді ми отримаємо готову секцію, яка до того є адаптованою під різні пристрої.

Також ми додаємо більше інтерактивності завдяки JS. Нам потрібно, щоб кожного разу на екрані з’являлася випадкова цитата. Тому ми створюємо масив із цитатами, генеруємо рандомне число в діапазоні довжини масиву та додаємо цю цитату на екран через ідентифікатор.

```
const quotesArray = [
    '"Reading a book is like having a conversation with the best men of the past centuries" Descartes',
    'Books are your best friends. You can turn to them in all the difficult moments of life. They will never betray you." A. Dode',
    '"The mind was not created by books, but books were created by the mind" G. Skovoroda',
]
const mainQuote = document.querySelector(".hero__text-decoration")
let randomQuotesIndex = Math.floor(Math.random() * quotesArray.length)
if (mainQuote) mainQuote.innerText = quotesArray[randomQuotesIndex]
```

Рис. . Додавання рандомної цитати із масиву на екран

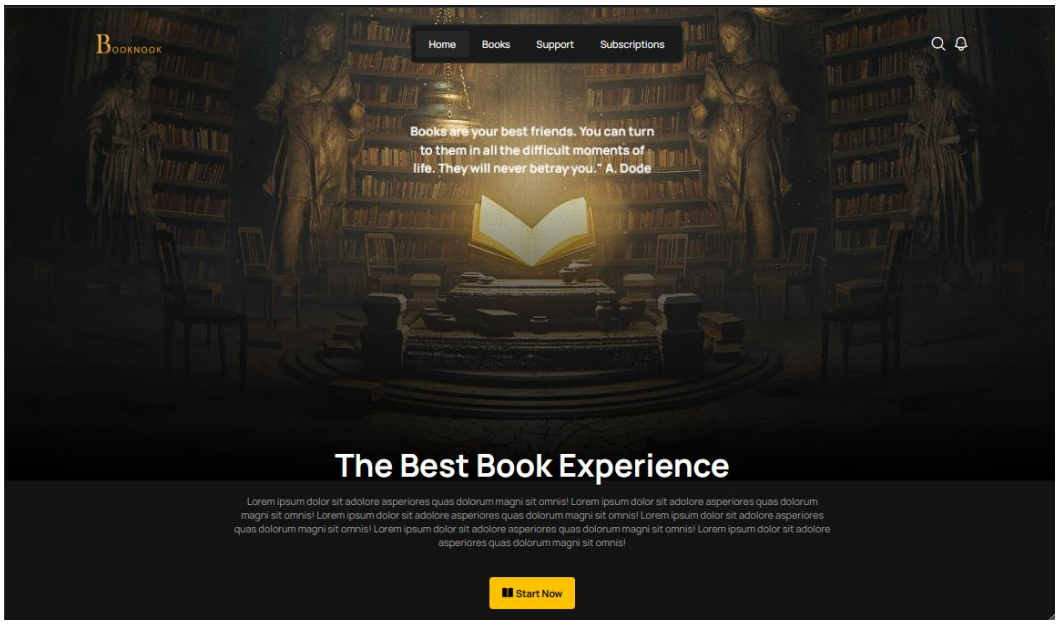


Рис. . Зверстана секція Hero

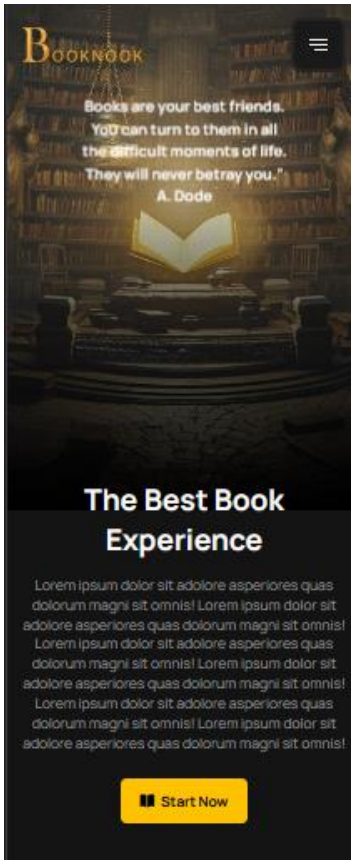


Рис. . Адаптована секція Hero

Отже, дана секція повністю готова та адаптована, навіть якщо контент буде змінюватися це ніяк не вплине на структуру сторінки.

Наступним елементом є карусель із жанрами книг. Для цього ми використаємо безкоштовну бібліотеку Swiper.js. Спочатку ми будуємо HTML. Так як цей об’єкт ми зробили раніше в _common.html ми просто копіюємо його та додаємо необхідні нам класи. Також обов’язково потрібно встановити у head посилання на стилі Swiper.js

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.css">
```

Та обов’язково підключаємо скрипти, що бібліотека працювала.

```
<script src="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.js"></script>
```

```html
<section id="homePageCategories" class="page__categories categories categories--margin">
    <div class="categories__container">
        <div id="homePageCategoriesOldPlace" class="titling titling--margin">···
        </div>
        <div id="homePageCategoriesNewPlace" class="categories__body-slider">
            <div id="homePageCategoriesSwiper" class="swiper categories__slider">
                <div id="categoriesWrapper" class="swiper-wrapper categories__wrapper">
                    <!-- <div class="swiper-slide categories__slide">···
                </div>
                <div class="swiper-pagination"></div>
            </div>
        </div>

    </div>
</section>
```

Рис. . Розмітка для каруселі жанрів книг.

Потім ми підключаємо карусель Swiper для нашої розмітки. Спочатку отримуємо доступ до блоку слайдера в HTML і при умові, що він є на сторінці, підключаємо до нього Swiper завдяки new Swiper (використовуємо ID через те, що подібних слайдерів на сторінці може бути багато і в усіх однакові класи). Потім ми налаштовуємо основні функції, такі як кількість відображення слайдів, навігацію та пагінацією, брейк-пойнти, де вказуємо зміну кількості слайдів при певній ширині екрану тощо.

```javascript
const categoriesHomePageSwiper = document.querySelector("#homePageCategoriesSwiper")
if (categoriesHomePageSwiper) {
    new Swiper("#homePageCategoriesSwiper", {
        loop: true,
        pagination: {
            el: "#homePageCategoriesLines",
            clickable: true,
        },
        navigation: {
            nextEl: "#homePageNext",
            prevEl: "#homePagePrev",
        },
        slidesPerView: 5,
        spaceBetween: 30,
        speed: 600,
        keyboard: {
            enabled: true,
            onlyInViewport: false,
        },
        breakpoints: {
            320: {
                slidesPerView: 1.5,
                spaceBetween: 5,
            },
            500: {
                slidesPerView: 2,
                spaceBetween: 10,
            },
            600: {
                slidesPerView: 3,
                spaceBetween: 15,
            },
            830: {
                slidesPerView: 4,
                spaceBetween: 20,
            },
            1208: {
                slidesPerView: 5,
                spaceBetween: 30,
            },
        },
    })
}
```

Рис. . Налаштування слайдера для категорій книг

Після налаштування ми зробимо ще одну важливу річ. У середині swiper-wrapper мають бути слайди із нашими фото та текстом, але ми будемо додавати їх із файлу JSON. Це потрібно для того, щоб в майбутньому легко підключити базу даних до сайту.

Першим кроком буде створити цей JSON файл, який буде відтворювати дані із бази даних, міститиме заголовок, посилання на фото тощо. Він матиме такий вигляд:


```
[
  {
    "id": 1,
    "link": "#",
    "image": "img/page/movie-categories/novels.webp",
    "name": "Романи"
  },
  {
    "id": 2,
    "link": "genres.html",
    "image": "img/page/movie-categories/sciense-fiction.jpg",
    "name": "Фантастика"
  },
  {
    "id": 3,
    "link": "#",
    "image": "img/page/movie-categories/science.png",
    "name": "Наукова література"
  },
  {
    "id": 4,
    "link": "#",
    "image": "img/page/movie-categories/detective.webp",
    "name": "Детектив"
  },
  {
    "id": 5,
    "link": "#",
    "image": "img/page/movie-categories/horor.jpg",
    "name": "Горрор"
  },
  {
    "id": 6,
    "link": "#",
    "image": "img/page/movie-categories/fantasy.webp",
    "name": "Фнтезі"
  }
]
```


Рис. . JSON-файл для слайдера

Потім в нашому скрип-файлі ми виконуємо такі дії: завдяки асинхронному методу fetch отримуємо доступ до нашого файлу. Потім потрібно отримати доступ до блоку, де будуть усі слайди. Якщо цей блок існує ми переходимо далі і створюємо цикл перебору, в якому створюємо змінну, в якій буде міститись розмітка нашого слайду. Завдяки інтерполяції ми додаємо необхідні дані із поточного елемента в наш слайд і після цього додаємо розмітку в наш блок.

```
fetch("../backend/categories.json")
   .then(pesponce => pesponce.json())
   .then(json => {
      const categoriesWrapper = document.querySelector("#categoriesWrapper")
      if (categoriesWrapper) {
         json.forEach(elem => {
            let result = `
            <div class="swiper-slide categories__slide">
            <a href="${elem.link}" class="categories__body">
               <div class="categories__images">
                  <img src="${elem.image}" alt="image" class="categories__img">
               </div>
               <div class="categories__info">
                  <h5 class="categories__name">${elem.name}</h5>
                  <div class="categories__link _i-slide-right"></div>
               </div>
            </a>
            </div>
            `
            categoriesWrapper.innerHTML += result
         })
         initiatecategoriesHomePageSwiper()
      }
   })
```

Рис. . Асинхронне додавання слайдів на сторінку

Функція initiatecategoriesHomePageSwiper потрібна, щоб ініціалізувати налаштування слайдера, згадані раніше, після того, як фото із JSON будуть додані на сторінку.

Як результат отримаємо таку сторінку:

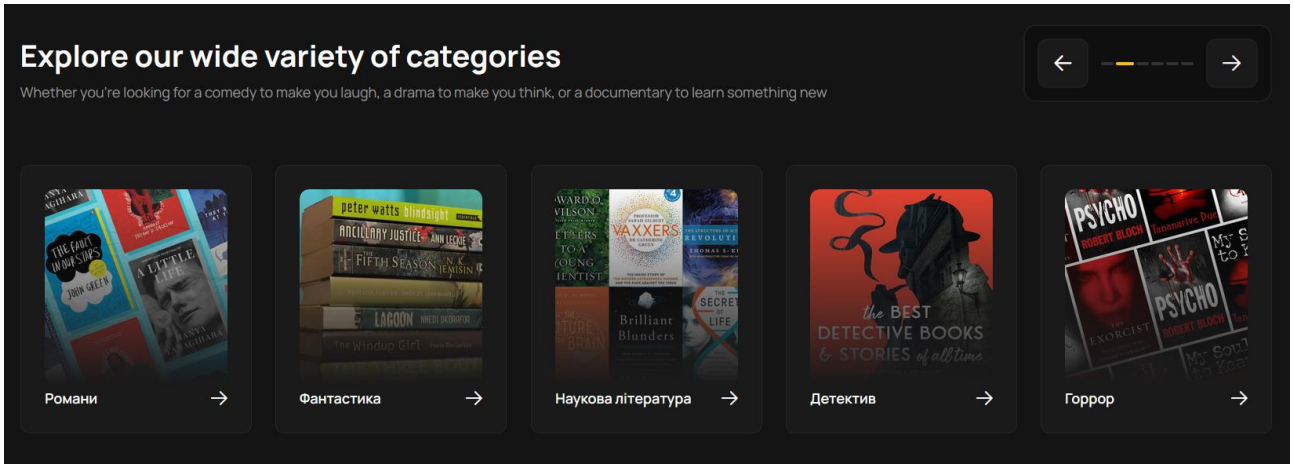


Рис. . Готовий слайдер

Потім завдяки match ми змінюємо розташування та стилі пагінації для усіх слайдерів сторінок:

```
const bookSectionLinesSwiperNavigation = document.querySelectorAll(".book-section__lines-swiper-navigation")
const bookSectionLinesSwiperNavigationOldPlace = document.querySelectorAll(".book-section__pagination")
const bookSectionLinesSwiperNavigationNewPlace = document.querySelectorAll(".book-section__block-slider")

if (window.innerWidth <= 1400) {
   for (let i = 0; i < bookSectionLinesSwiperNavigationNewPlace.length; i++) {
      bookSectionLinesSwiperNavigationNewPlace[i].appendChild(bookSectionLinesSwiperNavigation[i])
   }
}
if (bookSectionLinesSwiperNavigation && bookSectionLinesSwiperNavigationOldPlace && bookSectionLinesSwiperNavigationNewPlace) {
   const bookSliderBreaKPoint = `(max-width: 1400px)`
   const bookMatchMedia = window.matchMedia(bookSliderBreaKPoint)
   bookMatchMedia.addEventListener("change", e => {
      const isTrueForBook = e.matches
      if (isTrueForBook) {
         for (let i = 0; i < bookSectionLinesSwiperNavigationNewPlace.length; i++) {
            bookSectionLinesSwiperNavigationNewPlace[i].appendChild(bookSectionLinesSwiperNavigation[i])
         }
      } else {
         for (let i = 0; i < bookSectionLinesSwiperNavigationOldPlace.length; i++) {
            bookSectionLinesSwiperNavigationOldPlace[i].appendChild(bookSectionLinesSwiperNavigation[i])
         }
      }
   })
}
```

Рис. . Перенесення пагінації для слайдерів

Після цього усі слайдери будуть налаштовані та адаптовані. На екрані телефона слайдер матиме такий вигляд:

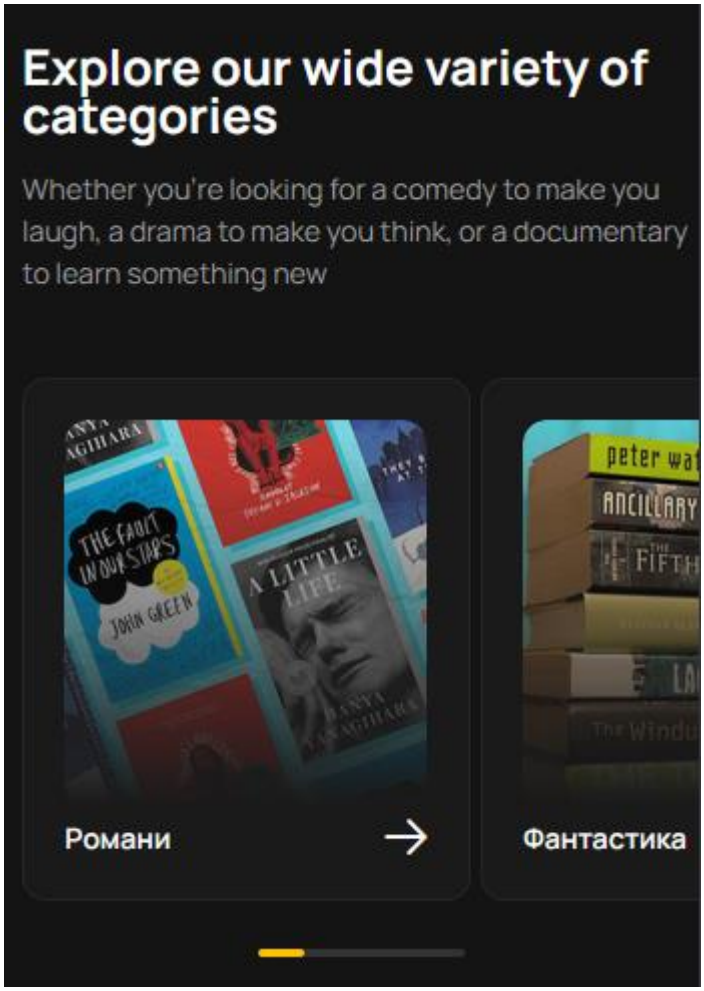
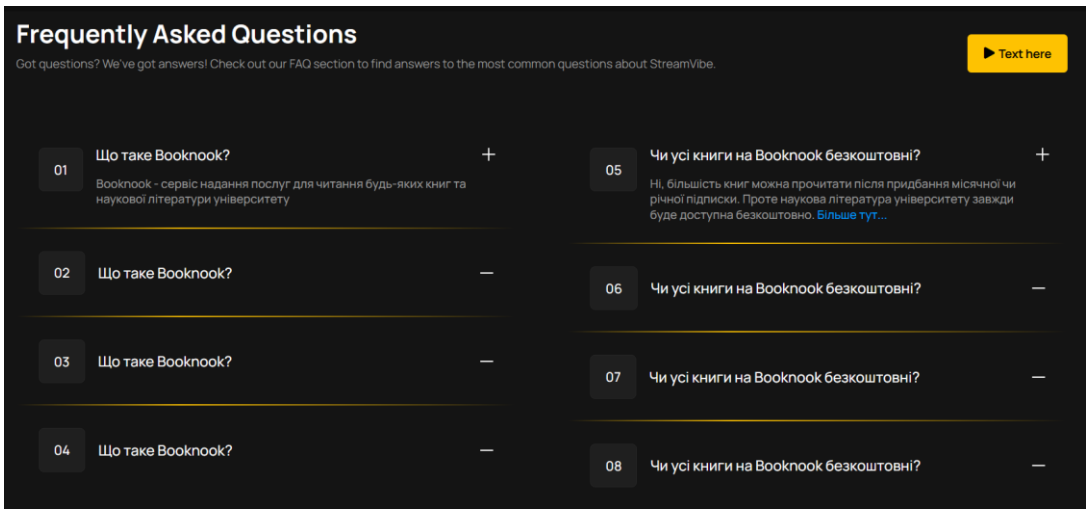
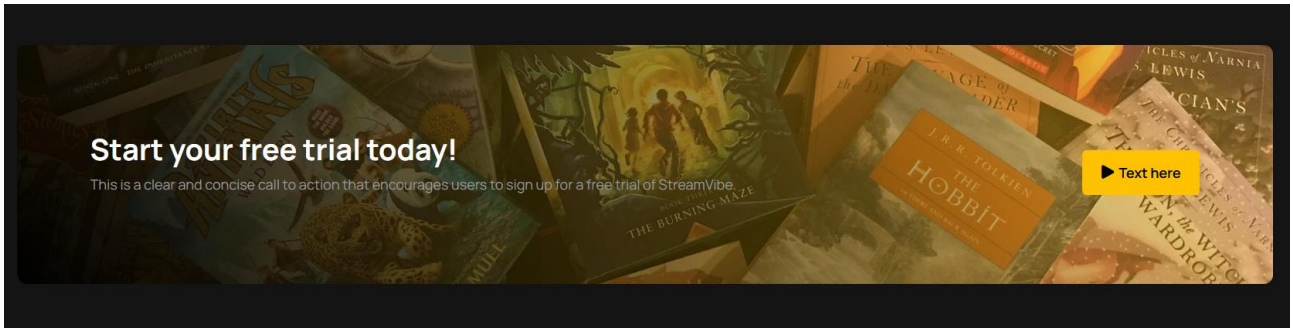


Рис. . Вигляд слайдера на екрані смартфона

Наступним важливим елементом на цій сторінці є секція «відповіді на запитання». Заголовок у нас вже готовий у _common.html, а саме тіло буде побудоване завдяки двом колонкам із списків ul/li.





3.1.6 Верстка сторінки «Каталог»

3.1.7 Верстка сторінки «Жанри»

Сторінка жанри складається в основному із слайдері, де кожен елемент – це посилання на іншу сторінку із книгами. Ці слайдери вже зверстані та зроблені раніше, необхідно лише додати їх на сторінку та змінити вміст.

Для кожного слайдера ми додаємо унікальний ідентифікатор для того, щоб підключити бібліотеку Swiper. Завдяки цьому кожен слайдер працює окремо і не впливає на роботу іншого.

Додавши брейк-пойнти ми адаптуємо наші слайдери до різних розширень екрану.

Найважчим етапом у верстці цієї сторінки буде адаптувати лінію навігації. Тут необхідно використати JavaScript для перенесення блоків у інше місце.

….

Інші стилі можна налаштувати через медіа запити CSS.

….

Також при зміні розміру екрану можна помітити, що слайдери все одно займають багато місця і з’являється дуже велика вертикальна прокрутка. Тому для вирішення цієї проблеми можна використати «таби».

Таби – це…

Фото

Таким чином ми маємо дві різні частини.

3.1.8 Верстка сторінки «Про книгу»

Дана сторінка містить детальний опис самої книги чи документа. Тут є складна структура із блоками, де є текст-опис, відгуки до цієї книги, а також інформація про рік, автора і т.д

3.1.9 Створення сторінки «Читати книгу»

Цей HTML-документ матиме на меті лише одне – завантажувати на екран документи або книги. Зараз така реалізація не розглядається в рамках проекту, але пізніше, після підключення бази даних, можна налаштувати код для рендерингу книг та документів.

3.1.10 Верстка сторінки «Підтримка»

На цій сторінці має бути

3.1.3 Додавання інтерактивності за допомогою JavaScript.

3.2 Перевірка роботи сайту на різних пристроях і браузерах.

3.3 Виявлення і виправлення можливих помилок у відображенні та функціональності.

3.4 Покращення швидкодії сайту.

3.5 Оптимізація для SEO.

3.6 Вдосконалення адаптивності до різних пристроїв і роздільних здатностей екрану.