

Geekbrains

Создание адаптивного веб-сайта магазина картин.

Программа:
Frontend-разработчик
Ульянова Д.А.

Москва
2024

Цель: Создать адаптивный веб-сайт магазина картин (фронтенд часть) используя для верстки макет Figma и разместить веб-сайт в сети интернет

Задачи:

1. Рассмотреть основные подходы к созданию веб-сайтов, выбору оптимальных инструментов и методологии для создания веб-сайта
2. Выбрать макет для верстки сайта в формате Figma находящийся в свободном доступе в сети интернет
3. Выполнить верстку сайта – создать структуру веб-страницы и сделать ее стилизацию
4. Придать сайту интерактивность используя возможности CSS3 и JavaScript
5. Проверить веб-сайт на ошибки
6. Разместить веб-сайт в сети интернет
7. При выполнении проекта получить практический опыт по созданию веб-сайта, развить и закрепить знания полученных в ходе обучения в GeekBrains, определить направления для дальнейшего обучения и практики

Инструменты: HTML, CSS, SASS, JavaScript, Figma, Visual Studio Code, Git, GitHub и другие

ОГЛАВЛЕНИЕ

Введение	1
Глава 1. Основы создания современных веб-сайтов.....	4
1.1 Определение веб-сайта и история создания.....	4
1.2 Классификация веб-сайтов	5
1.3 Подходы и способы создания веб-сайтов.....	9
1.4 Современные программные средства для создания веб-сайтов.....	13
1.4.1 Фронтенд и бэкенд сайта.....	13
1.4.2 Обзор программных средств фронтенд разработки	14
1.5 Этапы выполнения проектов по созданию веб-сайтов	21
Глава 2 Практическая реализации задачи по созданию адаптивного веб-сайта.....	24
2.1 Инструменты, применяемые при создании сайта	24
2.2 Выбор макета Figma.....	27
2.3 Верстка главной страницы сайта	29
2.3.1 Файловая система проекта	29
2.3.2 Создание структуры страницы.....	30
2.3.3 Семантика HTML страницы.....	32
2.4 Стилизация главной страницы сайта	32
2.4.1 Методология БЭМ для верстки и стилизации сайта	33
2.4.2 Адаптивность - отзывчивость сайта	34
2.4.3 Использование изображений.....	36
2.4.4 Загрузка и подключение шрифтов.....	37
2.4.5 Нормализация настроек	38
2.5 Создание интерактивных элементов главной страницы сайта.....	40
2.6 Разработка внутренних страниц сайта	51
2.7 Проверка веб-сайта на ошибки.....	62
2.8 Загрузка веб-сайта в сеть интернет	67
Заключение.....	68
Литература.....	69
Приложения.....	71

ВВЕДЕНИЕ

На сегодняшний день интернет является одним из основных источников информации для большинства населения планеты. Согласно отчету Global Digital 2023, выпущенного аналитической компанией Meltwater, в мире насчитывается 5,16 миллиарда пользователей интернета, что составляет 64,4% мирового населения.

За последний год количество интернет-пользователей выросло на 1,9%.

Текущие тенденции прироста количества интернет-пользователей показывают, что к концу 2023 г. почти 2/3 населения мира будут подключены к интернету.

Хотя поведенческие привычки пользователей сети меняются, «поиск информации» по-прежнему является основной причиной использования интернета - 57,8% интернет-пользователей трудоспособного возраста обращаются к онлайн-ресурсам в поисках информации по данным исследований компании GWI.

При этом поиск товаров в интернете и покупки онлайн стали повседневной реальностью расширившись от привычных категорий продукции, покупаемой онлайн, до продуктов питания и товаров повседневного спроса. Согласно данным компании Statista 17,1% глобальных расходов в ритейле за последний год приходится на онлайн торговлю. По прогнозам аналитиков GWI и Statista, в ближайшие годы рост доли e-commerce сохранится.

В этой связи активное присутствие компании в интернете является чрезвычайно важным практически для любого вида бизнеса.

Активное позиционирование в социальных сетях и современный веб-сайт являются неотъемлемым фактором успеха как для малого бизнеса, так и для глобальных корпораций.

Веб-сайт — это окно в мир для бизнеса, которое позволяет рассказать о компании и продуктах широкому кругу потенциальных клиентов, наладить продажу товаров в случае создания онлайн магазина и многое другое.

Чтобы это окно полностью открылось современный веб-сайт должен соответствовать ряду критериев прежде всего быть адаптивным, то есть корректно работать на различных устройствах - десктопе, ноутбуке, планшете и смартфоне.

Ожидается, что время использования мобильных устройств для выхода интернет продолжит расти, однако компьютеры останутся важной частью цифрового пространства на ближайшие годы.

Эти факторы определили выбор темы моей дипломной работы – “Создание адаптивного веб-сайта магазина”, что является чрезвычайно актуальным с учетом текущей ситуации и прогнозируемыми тенденциями развития цифрового пространства в России и мире.

Кроме того, работа над этим проектом позволяет погрузиться в типичную рабочую задачу по созданию веб-сайта на основе макета, полученного от дизайнера, что является одной из востребованных и широко распространённых задач фронтенд разработчика.

Таким образом, целью данного проекта является разработка веб-сайта на основе выбранного макета, в котором реализуются наиболее оптимальные инструменты и современные подходы, а именно верстка, стилизация, придание интерактивности и загрузка в сети интернет.

В связи с тем, что выбранный макет включает в себя только главную страницу сайта, в ходе выполнения работы было принято решение о расширении объема проекта и создании ряда внутренних страниц сайта на основе собственного дизайна. Благодаря этому удалось придать логическую завершенность разделам главной страницы и реализовать значительно больший функционал веб-сайта.

Данный проект также дал возможность смоделировать работу в команде над реальным проектом, выполняемую в отсутствии четкого технического задания и ограниченную по срокам, с чем нередко можно столкнуться в реальной жизни.

Все это позволило раскрыть в рамках одного проекта все необходимые практические навыки по разработке веб-сайтов востребованные на рынке фронтенд разработок.

ГЛАВА 1. ОСНОВЫ СОЗДАНИЯ СОВРЕМЕННЫХ ВЕБ-САЙТОВ

1.1 Определение веб-сайта и история создания

Согласно определению, данному на специализированном сайте для разработчиков MDN Web Docs:

“Веб-сайт это - это коллекция страниц, связанных между собой какими-либо способами (включая их связи с иными ресурсами), которые доступны под единым доменным именем.”

При этом страница или веб-страница - это документ, написанный на языке HTML, который может содержать текстовую и мультимедийную информацию (изображение, музыка, видео).

Веб-страницы отображаются на экране компьютере посредством браузера – программы для поиска и отображения веб-страниц.

Каждая страница в сети интернет имеет свой уникальный адрес.

Для доступа к нужной странице необходимо набрать ее адрес в адресной строке браузера.

- История создания

Первый в мире веб-сайт info.cern.ch был создан британским исследователем Тимом Бернерсом-Ли в 1991 г. для проекта «World Wide Web» CERN (Европейский центр ядерных исследований).



Tim Berners-Lee, pictured at CERN (image: CERN)

На этом сайте были сформулированы основные определения Интернет, размещены инструкции по установке веб-сервера, использования браузера и т.п.

Фактически это был первый в мире интернет-каталог, так как он содержал список ссылок на другие сайты.

Однако, важно отметить, что теоретические основы веба были заложены гораздо раньше различными учеными в 50-60 годах двадцатого века.



Скриншот восстановленной версии первого веб-сайта
(Изображение ЦЕРН)

1.2 Классификация веб-сайтов

С момента создания первого сайта интернет продолжает стремительно развиваться и сегодня в нем представлено огромное количество самых разнообразных веб-сайтов.

Хотя не существует единой классификации веб-сайтов их можно разделить по ряду характерных признаков таких как:

- Размер
- Применяемая технология
- Назначение



По размеру

По размеру сайты бывают одностраничными и многостраничными.

Типичными примерами одностраничного сайта является landing page или сайт-визитка, размещающий всю информацию на одной странице.

Одностраничные сайты позволяют эффективно представить пользователю информацию об определенном продукте, рассказать о услугах специалиста или компании.

Многостраничные сайты могут иметь разную структуру, количество страниц, тематическую организацию и схему внутренних связей. Общим для многостраничных сайтов является наличие главной страницы, имеющей связи со всеми тематическими разделами, которые в свою очередь могут иметь собственные главные страницы и подразделы со страницами следующего уровня. Количество страниц многостраничного сайта определяется общим объемом и характером материалов веб-сайта.

По технологии

По применяемой технологии сайты можно разделить на статические, динамические и flash-сайты.

Статические и динамические сайты являются двумя основными типами веб-сайтов. Они отличаются способом создания, обработкой контента и возможностями для пользователей. Выбор между статическим и динамическим сайтом зависит от целей и требований к проекту.

Статические сайты

Статический сайт – это сайт, который состоит из фиксированных веб-страниц с постоянным содержанием. Каждая страница представляет собой отдельный HTML-файл, который хранится на сервере и отображается в браузере пользователя без изменений.

Статические сайты хорошо подходят для создания небольших сайтов с постоянным контентом, а также там, где имеется сложный дизайн каждой из

страниц и приоритетна скорость загрузки - лендинг-страницы, сайты визиток, каталогов продукции.

Достоинства статических сайтов:

- Простота создания и поддержки
- Быстрая загрузка страниц
- Не требуют сложных серверных настроек
- Не зависят от баз данных и серверных языков программирования

Недостатки статических сайтов:

- Более сложное обновление
- Более сложное масштабирование
- Более затратное внесения изменений

Динамические сайты

Динамический сайт – это сайт, который генерирует контент в реальном времени в зависимости от взаимодействия пользователя с сайтом или настроек пользователя. Такие сайты используют серверные языки программирования и базы данных для хранения и обработки информации.

Динамические сайты предназначены для создания сложных веб-приложений и предоставляют пользователям больше возможностей для взаимодействия.

Это особенно актуально для сайтов социальных сетей, блогов и интернет магазинов.

Достоинства динамических сайтов:

- Гибкость и масштабируемость
- Возможность создания сложных веб-приложений
- Больше возможностей для взаимодействия с пользователем
- Возможность использование CMS
- Низкая стоимость текущего обслуживания

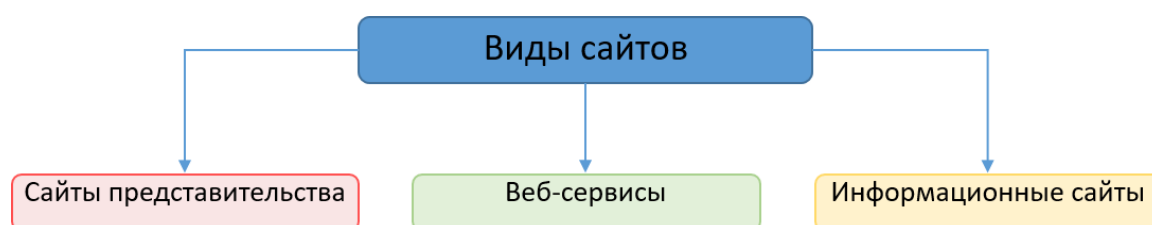
Недостатки динамических сайтов.

- Ограниченные возможности для дизайна
- Более высокие затраты если используются нешаблонные решения

По назначению

По назначению сайты можно разделить на три основные группы:

- Сайты представительства
- Информационные сайты
- Веб-сервисы



Примерами информационных сайтов являются – различные тематические сайты, сайты новостей, электронные библиотеки, энциклопедии, словари, хранилища фалов и др.

К сайтам представительствам можно отнести широкую гамму корпоративных и коммерческих сайтов - сайты компаний, интернет магазины, а также сайты представляющие государственных учреждения, сайты визитки, промо-сайты и тд.

Сайты веб сервисов включают - сайты поисковых систем, почтовые сервисы, сайты перевод текста, доски объявлений, видео-хостинги, системы электронных платежей, социальные сети и другие.

1.3 Подходы и способы создания веб-сайтов

По способу создания веб-сайтов можно выделить три подхода:

1. Разработка с применением конструктора сайта
2. Разработка на CMS-системе
3. Разработка сайта с использованием языка разметки HTML и языков программирования

Разработка с применением конструктора

Конструктор – онлайн сервис, позволяющий создать сайт на основе готовых шаблонов, которые предоставляет платформа. При таком подходе возможно создать простой сайт имея минимальные знания или вообще без знаний о веб-разработке. Разработать сложное веб-приложение или интернет-магазин на конструкторе невозможно. Наиболее рационально использовать конструктор сайтов для простых сайтов визиток или лэндинг страниц.

Плюсы:

- Низкая цена
- Простота использования
- Готовые стандартные модули и легкость использования

Минусы:

- Скрытые затраты, размещение на хостинге разработчика конструктора, электронная почта с именем домена и др
- Домен третьего или более высокого уровня
- Тяжеловесность сайта
- Ограничения по возможности SEO

Существует большое количество конструкторов сайтов, некоторые из наиболее популярных - Tilda Publishing, Nethouse и Wix.

Разработка на CMS-системе

CMS (Content management system) – система управления содержимым - набор базовых и вспомогательных инструментов, позволяющих создать веб-сайт и

обеспечить его работу, обновление содержимого и взаимодействие с пользователями сайта. Разработка осуществляется через панель управления. CMS позволяет реализовать гораздо более значительные проекты, по сравнению с конструкторами сайтов, например, сайт интернет-магазина или многостраничный корпоративный сайт.

Плюсы:

- Практически все CMS изначально бесплатны. Имеется большое количество готовых шаблонов сайтов в открытом доступе
- Легкость управления контентом
- Множество готовых решений – моделей, плагинов и дополнений

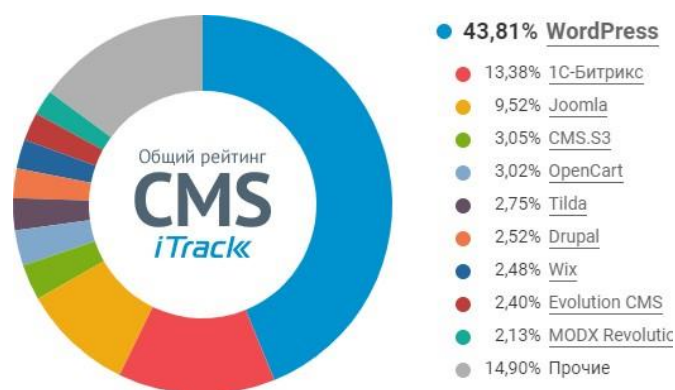
Минусы:

- Уязвимость сайта от атак и проникновения
- Требуется знание основ верстки и программирования в случае не стандартных решений
- Возможные сложности с миграцией между хостингами
- Платный дополнительный контент, более затратный по сравнению с конструкторами сайтов
- Значительный рост затрат при усложнении проекта

Самой известной и широко используемой CMS на сегодняшний день является WordPress.

По данным исследования компании iTrack доля WordPress среди CMS в

доменной зоне ru составляет более 43%. На WordPress реализовано более 500 тыс самых разнообразных проектов от простых интернет сайтов до крупных информационных порталов.



Данные <https://itrack.ru/> март 2021 г.

Следующими по популярности из условно бесплатных CMS идут Joomla и OpenCart.

Joomla считается универсальной с точки зрения категории сайта и подходит для создания как простых, так и относительно сложных проектов.

OpenCart - пользователи отмечают удобство использования этой системы особенно для создания интернет-магазинов.

Также стоит отметить платформу CMS Bitrix оптимизированную под работу с системой 1С.

При выборе между конструктором сайтов и системой CMS среди прочих факторов необходимо учитывать, что при использовании CMS сайт может быть размещен на сервере клиента, то есть клиент владеет всем проектом, тогда как сайт сделанный в конструкторе находится на хостинге разработчике платформы, с отсутствием возможности доступа к исходному коду.

Создание сайта на языках программирования

Использование языков программирования позволяет создавать проекты любой сложности с учетом индивидуальных требований клиента.

Написания сайта на языках программирования можно сравнить с пошивом одежды на заказ, когда будут учтены все особенности фигуры и стилевые предпочтения, но процесс потребует больше времени и усилий.

Плюсы:

- Возможность создания уникального, не шаблонного решения максимально оптимизированного для решения конкретных бизнес задач клиента
- Большие возможности продвижения по сравнению с сайтами, сделанными с помощью конструкторов и CMS систем
- Индивидуальный дизайн способный обеспечить максимальную эффективность UI/UX

Минусы:

- Более высокая цена простых сайтов по сравнению с CMS и конструкторами
- Требуется широкий спектр знаний - программирования, архитектуры веб-приложений, алгоритмов, структур данных и многого другого, что ограничивает возможности самостоятельной разработки и требует привлечения специалистов
- Сроки разработки простых сайтов больше по сравнению CMS и конструкторами.

1.4 Современные программные средства для создания веб-сайтов

1.4.1 Фронтенд и бэкенд сайта

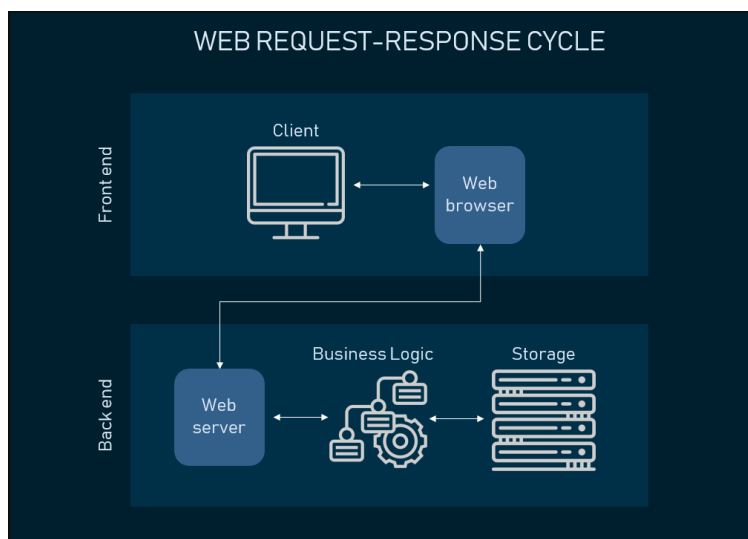
В настоящее время разработку сайта принято разделять на две составляющие — фронтенд и бэкенд.

Фронтенд веб-сайта — это все, что пользователь видит и с чем может взаимодействовать при помощи браузера. Создание этой визуальной части называется фронтенд-разработкой.

Для разработки фронтенда в качестве базовых инструментов используются: HTML (создание базовой структуры страниц и контента), CSS (стилирование внешнего вида) и JavaScript (добавление интерактивности).

Бэкенд — это серверная часть веб-приложения, скрытая от глаз пользователя. Это понятие включает в себя серверы, на которых расположены веб-страницы и определенную логику управления функционалом и процессами работы сайта.

Бэкенд разрабатывается на другом технологическом стеке, включающем Java, PHP, Ruby, C# и иные программные средства.



Цикл запрос-ответ

1.4.2 Обзор программных средств фронтенд разработки

Как было отмечено выше базовыми инструментами фронтенд разработки являются HTML, CSS и JavaScript.

Сайт MDN Web Docs дает следующие определения для этих средств:

- HTML (Hypertext Markup Language) - это код, который используется для структурирования и отображения веб-страницы и её контента. Контент может быть структурирован внутри параграфов, маркированных списков или с использованием изображений и таблиц данных. HTML не является языком программирования. Это язык разметки, сообщающий браузеру, как отображать веб-страницы.
- CSS (Cascading Style Sheets) - это код, который используется для стилизации веб-страниц. CSS также не является языком программирования. Это язык таблицы стилей, позволяющий применять стили выборочно к элементам в документах HTML.
- JavaScript - это динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах.



JavaScript был создан известным программистом Brendan Eich во время его работы в компании Netscape. Первая версия JavaScript вышла в 1995 и с тех пор язык постоянно развивается.

JavaScript стандартизован ассоциацией ECMA International. Стандартизированная версия называется ECMAScript, описывается стандартом ECMA-262.

Стандарт ежегодно обновляется при этом одной из фундаментальных версий стандарта является ECMAScript 6 (ES-2015) потому что она внесла ряд существенных изменений, таких как:

- Объявление переменных с помощью `let` и `const`
- Стрелочные функции
- Промисы

- Шаблонные литералы
- Spread/Rest синтаксис
- Деструктуризация
- Цикл for...of
- Новые структуры данных Map и Set
- Параметры по умолчанию в функциях
- Новые методы объекта
- и ряд других

Важными вехами в развитии веба также стали выход стандарта HTML5 и CSS3.



Одной из основных целей разработки HTML5 являлось улучшение поддержки мультимедийных технологий, читаемости кода человеком и машиной с сохранением обратной совместимости.

Среди новшеств, появившихся в HTML5 можно отметить следующие:

- Новые элементы мультимедиа <video>, <audio>
- Улучшенные формы
- Улучшение разметки документа и новые семантические теги
- Переопределение и стандартизация ряда элементов
- Математические формулы
- Файлы svg
- Обработка ошибок
- Геолокация
- Управление кэшем для офлайн-работы
- Элемент <canvas> для непосредственного метода рисования
- Ряд новых API
- и другие



Выход стандарта CSS3 предоставил много новых возможностей по стилизации html документов, прежде всего это создание анимированных элементов без использования JavaScript, а также многое другое:

- Внедрение переменных
- Расчет значений с использованием calc()
- Расширение возможностей селекторов – псевдоэлементы и псевдоклассы
- Плавность прокрутки
- Закрепление элементов на странице position: sticky
- Подключение сторонних шрифтов
- Добавление теней к тексту
- Рамки для форматирования границ элемента
- Улучшение возможностей адаптивной верстки:
 - Медиазапросы
 - Новые единицы измерения vh, vw, em, rem
 - Новые адаптивные возможности flexbox и grid layout
- Трансформация и анимации элементов

Кроме базовых средств фронтенд разработки, есть широкий спектр дополнительного инструментария - диспетчеры пакетов, CSS-препроцессоры, фреймворки, библиотеки и многое другое.

CSS-фреймворки

CSS-фреймворки – готовые к использованию стандартизованные CSS и HTML компоненты, позволяющие упростить и ускорить разработку, а также минимизировать ошибки на этапе верстки и стилизации.

CSS-библиотеки обычно подключаются в виде внешнего css-файла в теге head html страницы.

Плюсы:

- Увеличивают скорость разработки

- Обеспечивают кроссбраузерность и адаптивность
- Применение чистых и масштабируемых шаблонов
- Единообразие шаблонных решений улучшают взаимодействие при работе команде

Минусы:

- Увеличивают “вес” проекта не используемым кодом
- Варианты дизайна ограничены стандартами применяемого фреймворка
- Создают “многоклассие” стилей на одном элементе

CSS-фреймворки отличаются по объему функциональности и часто классифицируются по этому признаку как полнофункциональные и легкие, имеющие только специализированные инструменты.

CSS-фреймворков также используют разные подходы к обеспечению кроссбраузерности применяя либо сброс стилей - reset.css или нормализацию Normalize.css.

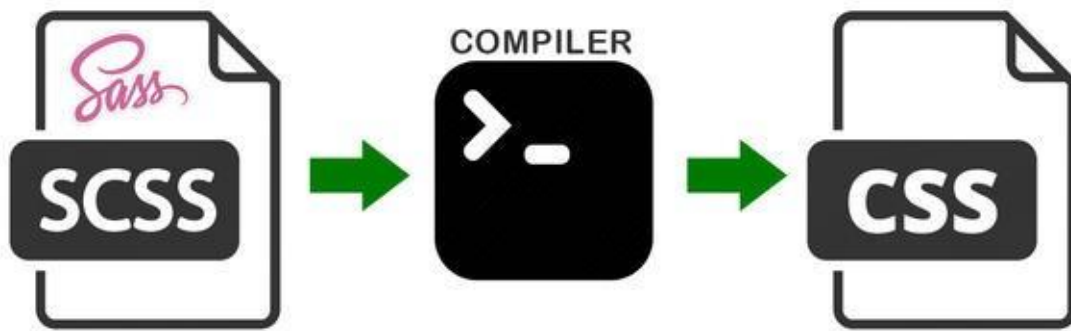
Существует большое разнообразие css-фрэймворков, наиболее популярными являются Bootstrap, Materialize CSS, Foundation и ряд других.

CSS препроцессоры

CSS препроцессоры - это программы, которые имеют собственные синтаксические конструкции расширяющие возможности CSS, улучшающие чистоту кода и уменьшающие его дублирование.

Препроцессоры позволяют при стилизации html-страниц использовать методы и конструкции характерные языкам программирования такие как: вложенность, наследование, переменные, циклы, примеси и другие.

Код написанный на препроцессорах схож с кодом CSS, но на для использования в браузере его необходимо преобразовать в чистый CSS с помощью компилятора.



Наиболее широко используемые CSS препроцессоров:

- SASS
- LESS
- Stylus
- PostCSS

Выбор препроцессора определяются особенностями конкретного проекта, личными предпочтениях, а также зависит от требований по интеграции с другими программными средствами, задействованными в проекте.

JavaScript фреймворки и библиотеки

Фреймворки и библиотеки JavaScript позволяют сделать процесс разработки веб-приложений более быстрым и эффективным. Они обладают широкой функциональностью и обеспечивают фронтенд разработчиков универсальными инструментами для решения разнообразных задач:

- Оптимизация работы с DOM
- Обеспечение предсказуемости и удобства поддержки приложений
- Облегчение работы программистов по тестированию, анализу правильности и лаконичности написания кода благодаря наличию экосистемы пользователей, создавших широкий спектр готовых решений
- Масштабируемость кода и улучшенное взаимодействие частей приложения друг с другом за счет компонентного подхода
- Упрощение создания маршрутизации

Все эти возможности фреймворков JavaScript лежат в основе их популярности. Много современных веб-приложений, включая ряд широко известных веб-сайтов сделаны с использованием фреймворков.

Не смотря на очевидные преимущества фреймворки не являются панацеей для решения любых проблем. Необходимо оценивать целесообразность их использования в каждой конкретной ситуации учитывая следующие факторы:

- Индивидуальные требования проекта
- Наличие знаний у специалиста и понимание особенностей работы фреймворка
- Готовность и способность всей команды разработчиков работать с определенным фреймворком
- Рациональность использования в простых проектах с небольшим объемом интерактивности

Фреймворки начали активно развиваться начиная с 2010 год конкурируя друг с другом за популярность среди разработчиков. В настоящий момент можно выделить следующие фреймворки входящие в “большую четверку”:

- Angular — фреймворк с открытым исходным кодом, созданный Google и сообществом частных лиц и компаний. Angular был официально выпущен 14 сентября 2016 года. Это компонентный фреймворк использующий декларативные HTML-шаблоны, которые компилятор преобразует в оптимизированные инструкции JavaScript во время сборки. Angular использует TypeScript.
- React - выпущен Facebook в 2013 году. До этого он продолжительное время использовался в компании при работе над внутренними задачами. Технически React сам по себе не является фреймворком; это библиотека для рендеринга UI компонентов, но так как он зачастую используется совместно с ReactDOM и React Native для создания веб и мобильных приложений, то он рассматривается в качестве фреймворка.
- Vue выпущен в 2014 разработчиком Evan You, который работал над созданием AngularJS. Это самый молодой фреймворк из большой четверки

при этом его популярность в последнее время очень возросла. Vue, как и AngularJS, расширяет HTML собственным кодом. Кроме того, он основан на современном стандарте JavaScript.

- Ember был выпущен в декабре 2011 года. На сегодняшний день он меньше используется по сравнению с React и Vue, но продолжает сохранять определенную популярность благодаря своей стабильности, рациональным принципам кодирования и поддержке сообщества.

Хотя библиотеки и фреймворки во многом схожи между ними есть отличия. Библиотека это хранилище фрагментов кода, который может быть использован разработчиком в приложении для решение определенной задачи или ограниченного набора задач.

Фреймворк создает каркас приложения на основе которого выстраивается вся функциональность приложения.

Приложение может быть создано, как только с использованием фреймворка, так и на одних библиотеках, но наиболее часто используется комбинирование возможностей обоих инструментов.

1.5 Этапы выполнения проектов по созданию веб-сайтов

В общем случае можно выделить четыре основных этапа разработки веб-сайта:

1. Подготовка
2. Проектирование
3. Разработка
4. Поддержка



На этапе подготовки определяются цели и задачи создания веб-сайта, исследуется целевой рынок, потребности потенциальных клиентов, деятельность конкурентов, определяется концепция сайта включая основной функционал и ассортимент продукции, которые будут представлено на сайте.

Результатом этапа подготовки становится техническое задание для проектирования веб-сайта.

На этапе проектирования разрабатывается архитектура и дизайн сайта. В этой работе участвует дизайнер, которые создает внешний вид сайта, также могут быть привлечены SEO-специалист и контент менеджер для оптимизации структуры сайта и его наполнения эффективно работающей информацией.



На следующем этапе к работе приступают программисты, которые реализуют проект в виде программного кода. Фронтэнд разработчик готовит всю интерфейсную часть проекта, а Бэкэнд специалист выполняет функциональную часть сайта с использованием необходимых серверных технологий.

Важным шагом на этом этапе является всесторонне тестирование веб-сайта, которое выполняется специалистом по тестированию программного обеспечения. Этап разработки завершается размещением сайта на хостинге, обучением заказчика пользоваться готовым сайтом и сдачей проекта.

Этап поддержки включает техническую поддержку – исправление багов, добавление нового функционала, рекламу и продвижение сайта в сети интернет. Состав работ на каждом этапе, количество и виды необходимых специалистов, оптимальная модель и методология разработки, сроки выполнения зависят как от сложности проекта, так и ряда других факторов.

Существуют различные модели и методологии управления созданием IT продуктов, которые применяются и при разработке веб-сайтов.

Выбор оптимального подхода особенно при реализации масштабных проектов является важным вопросом, который может значительно повлиять на сроки и бюджет проекта.

Можно отметить следующие модели разработки ПО:

- Waterfall модель
- V-образная модель
- Спиральная модель
- Интерактивная или итерационная модель
- Гибкие модели

В части методологий управления IT проектами одним из передовых направлений является использование гибких методологии разработки, которые могут оптимальным решением в случае необходимости реализации сложных задач в условиях сжатых сроков, неопределенности технических и других аспектов на момент старта проекта.



ГЛАВА 2 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИИ ЗАДАЧИ ПО СОЗДАНИЮ АДАПТИВНОГО ВЕБ-САЙТА

2.1 Инструменты, применяемые при создании сайта



Для написания кода применяется редактор исходного кода Visual Studio Code (VS Code).

VS Code разработан компанией Microsoft, выпуск первой версии состоялся в 2015 г.

VS Code заслужил огромную популярность во всем мире и был выбран для работы над проектом благодаря следующим факторам:

- Бесплатный, кросс-платформенный редактор с открытым исходным кодом
- Встроенная поддержка HTML, CSS, JavaScript, Node.js и многих других языков программирования
- Возможность выбора и изменения языковых режимов
- Встроенный отладчик, поддерживающий Node.js
- Встроенная интеграция с Git
- Широкие возможности настройки интерфейса
- Доступная документация и обучающее видео
- Возможность установки разнообразных расширений - просто и быстро используя встроенный интерфейс

В VS Code был установлен ряд расширений, таких как форматтеры и сниппеты для CSS, JavaScript, HTML, компилятор и транспайлер SASS, Live Server позволяющих сделать работу с этими программами более удобной.



Для обеспечения надежного хранения и версирования проекта применена система контроля версий Git.

Git создан Линусом Торвальдсом для организации разработки ядра Linux. Первая версия Git выпущена в 2005 году. Сегодня Git является самой популярной системой контроля версиями и используется в огромном количестве проектов по разработке программного обеспечения по всему миру. Git отлично работает со всеми популярными операционными системами и средами разработки.

Visual Studio Code полностью интегрирован с Git и позволяет мгновенно видеть внесенные изменения прямо в редакторе.

Для работы Git могут использоваться различные Git-клиенты с графическим интерфейсом, например Git Kraken, GitHub Desktop, SourceTree или окно терминала.

При подготовке дипломного проекта все взаимодействие с Git осуществлялось через терминал VS Code. Также, как и VS Code, Git был установлен локально на ноутбук.



Для работы с удаленными репозиториями выбран сервис GitHub разработанный компанией Microsoft.

GitHub является крупнейшей веб-платформой для хостинга IT проектов и совместной разработки. На конец 2023 года этим сервисом пользуется более 100 миллионов разработчиков, более 4 миллионов организаций, которыми было создано более 330 миллионов репозиториях.

Владельцем сервиса является компания Microsoft.

Специально для дипломного проекта на GitHub создан новый публичный репозиторий. Этот репозиторий используется как в ходе работы над проектом, что позволяет поддерживать активными навыки использования Git и GitHub, так и для передачи проекта на проверку экспертам GeekBrains.

2.2 Выбор макета Figma



Figma

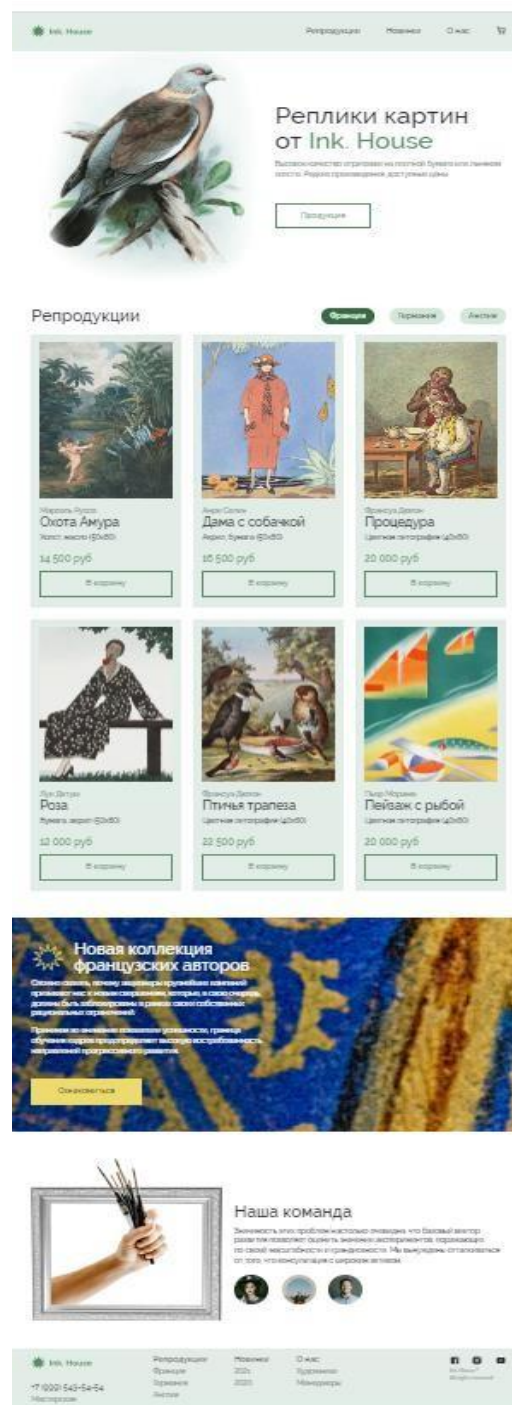
Для выполнения проекта выбран макет сайта, дизайн которого сделан в программе Figma.

Figma - это сервис для разработки и прототипирования веб-сайтов, мобильных приложений и других цифровых продуктов, позволяющий пользователь совместно работать над проектом в онлайн режиме.

Данный сервис начал развиваться в 2012 года и на сегодняшний день Figma стала одним из наиболее популярных графических редакторов в веб разработке в мире. Владелец Figma является компания Adobe.

Решение использовать этот макет принято по следующим причинам:

- Актуальность тематики - интернет магазины и лендинг страницы продаж являются востребованными продуктами в фронтенд разработке и прогнозируется что этот спрос сохранится в ближайшие годы
- Использование макета для разработки сайта позволяет смоделировать реальную рабочую ситуацию, когда разработчик получает макет от дизайнера и взаимодействует с ним в рамках одной команды
- Макет сайта включает дизайн для ПК, мобильной и планшетной версий сайта, что служит необходимой основой для создания качественного адаптивного веб-сайт. Адаптивность веб-приложений под



устройства различных форматов является очень важным для формирования положительного пользовательского опыта от взаимодействия с продуктом

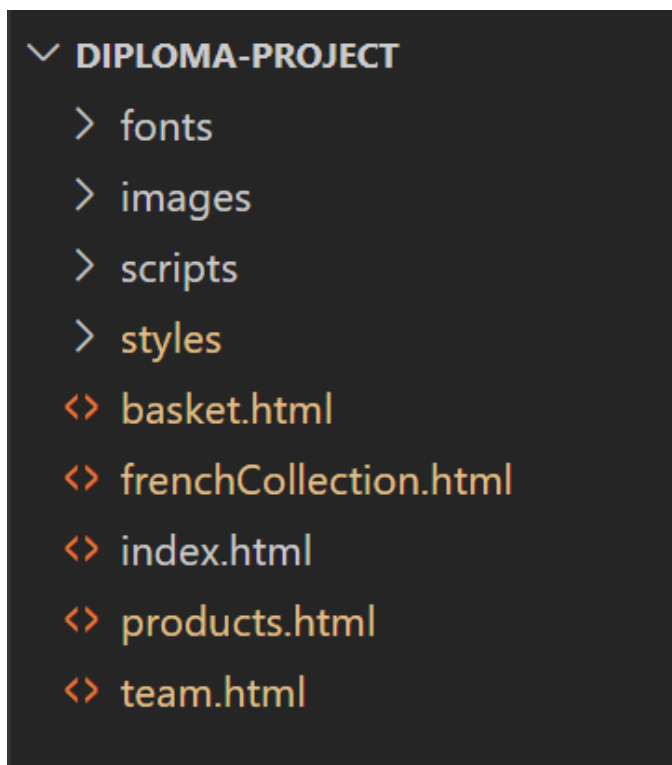
- Подходящий уровень сложности макета, который позволяет в рамках дипломной работы продемонстрировать имеющиеся и развить дополнительные навыки работы со многими ключевыми инструментами фронтенд разработки на этапе верстки, стилизации и анимирования сайта.
- Наличие бесплатных макетов Figma для свободного использования в сети интернет
- Бесплатная онлайн версия программы (осень 2023)
- Как было упомянуто во Введении, в ходе работы над проектом было принято решение дополнить главную страницу сайта, созданную по макету Figma рядом внутренних страниц собственной разработки. Благодаря этому веб-сайт получил логически более завершенную структуру со значительно расширенным функционалом.
- Подробная информация о дополнительных страницах сайта приводится в Главе №2, раздел 2.6 - Разработка внутренних страниц сайта.

Ссылка на макет в Figma - [House – Figma](#)

2.3 Верстка главной страницы сайта

2.3.1 Файловая система проекта

Для локальной работы с проектом на рабочем компьютере создана отдельная папка `Diploma_project`. В этой папке хранятся все файлы сайта согласно следующей структуре:



- папка `fonts` – нестандартные шрифты, используемые в проекте
- папка `images` – изображения
- папка `scripts` – файлы с JavaScript кодом для реализации интерактивных функций
- папка `styles` – CSS и SCSS файлы для стилизации контента
- файлы `index.html`, `basket.html*`, `frenchCollection.html*`, `products.html*`, `team.html*` – файлы с HTML кодом.

* - файлы относятся к внутренним страницам сайта

2.3.2 Создание структуры страницы

Используя редактор кода VS Code создан файл index.html – файл с HTML кодом и в нем задана базовая структуру главной страницы сайта.

Определен тип документа, язык содержимого, кодировка и заголовок страницы в браузере.

Остальные страницы сайта реализованы на основе аналогичного подхода.

```
<> index.html > ...
1  <!DOCTYPE html>
2  <html lang="ru">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>My Diploma - extended - Main page</title>
8      <link rel="stylesheet" href="styles/normalize.css">
9      <link rel="stylesheet" href="styles/style.css">
10     <script src="scripts/tabs.js" defer></script>
11     <script src="scripts/burger.js" defer></script>
12     <script src="scripts/basketCounter.js" defer></script>
13 </head>
```

В теге <head> подключен основной файл со стилями style.css и файлы с JavaScript кодом, прописаны относительные пути к этим файлам.

Разбиение на блоки:

```
15 <body>
16 >   <header class="header center">...
79   </header>
80   <main>
81 >     <section class="top center">...
91   </section>
92 >     <section class="catalog center">...
290   </section>
291 >     <section class="new-collection center">...
308   </section>
309 >     <section class="team center">...
324   </section>
325   </main>
326 >   <footer class="footer center">...
411   </footer>
412 </body>
```

Верстка макета была начата с десктопной версии сайта. Исходя из структуры макета в нем можно выделить следующие основные блоки:

- Шапку сайта <header>
- Основное содержимое сайта <main>
- Подвал сайта <footer>

Детализируя содержимое каждого из блоков в них выделены следующие основные компоненты:

Шапка

- Логотип сайта тег `<div class="logo">`
- Навигационное меню - тег `<nav class="header__nav">`

Основное содержимое расположено внутри тега `<main>`:

- Верхняя секция `<section class="top center">`
- Секция с каталогом репродукций `<section class="catalog center">`
- Секция с новинками `class="new-collection center">`
- Секция с наша команда `<section class="team center">`

Подвал

- Навигационное меню – тег `<nav class="footer__nav">`
- Логотип и контактная информация - `<div class="footer__contacts">`
- Иконки переходов на страницы в социальных сетях и резервация прав компании сгруппированы в теги `<div>` с соответствующими классами.

Далее каждый из компонентов разбивает на более мелкие элементы, такие как заголовки, изображения, текст, кнопки, ссылки, вспомогательные контейнеры и тд., в соответствие индивидуальным контентом каждого блока или секции.

2.3.3 Семантика HTML страницы

Используя семантический подход, то есть опору на смысловое предназначение компонентов и логическую структуру документа, заданы необходимые теги и прописаны в виде кода.

В проекте использованы следующие семантические элементы – section, main, header, footer, nav и другие.

Целью является оптимизация разметки страницы с точки зрения семантики, что является важным фактором при создании сайта так как позволяет:

- Сделать сайт более доступным для пользователей с ограниченными возможностями
- Обеспечить более высокое ранжирование в поисковой выдаче
- Делает код более понятным и структурированным, что облегчает командную разработку и последующую поддержку сайта

Кроме этого стратегия построения разметки страницы направлена на обеспечения максимальной отзывчивости и адаптивности сайта под различные типы мобильных устройств.

2.4 Стилизация главной страницы сайта

Стилизация html страницы выполнена с использованием препроцессора SASS. Для работы с SASS создан файл style.scss и установлены необходимые расширения в редакторе VS Code (*1. Live Sass Compiler 2. Sass 3. Live Server*).

В проекте используется следующий функционал SASS:

- Вложенность
- Фрагментирование
- Примеси
- Переменные
- Импорт

Стили в SASS написана в синтаксисе SCSS.

Переменные и миксины используемые в проекте располагаются в файле `_vars.scss`, который подключается к основному файлу со стилями с помощью директивы `@import 'vars';`

Пример миксина используемого при стилизации:

```
@mixin button {  
  font-size: 20px;  
  font-style: normal;  
  font-weight: 500;  
  line-height: normal;  
  transition: .3s;  
}
```

Стили, относящиеся к HTML коду, который повторно используются на внутренних страницах сайта, в частности шапка и подвал вынесены в отдельные файлы, подключаемые к основному файлу со стилями с помощью директивы `@import`.

2.4.1 Методология БЭМ для верстки и стилизации сайта

Определение стилей для элементов html выполнено с применением селекторов класса при этом название классов задаются на основе методологии БЭМ (Блок-Элемент-Модификатор).

Методологию БЭМ придумали и начали развивать в компании Яндекс с 2005 года для улучшения исполнения собственных проектов. Методология показала свою эффективность во множестве проектов как Яндекс, так и других компаний, и сейчас широко используется разработчиками по всему миру.



Блок - Логически и функционально независимый компонент страницы.

Элемент - Составная часть блока, которая не может использоваться в отрыве от него.

Модификатор - сущность определяющая внешний вид, состояние и поведение блока или элемента.

Положительные эффекты этой методологии проявляются в дипломном проекте:

- Наглядная (Понятная) структура проекта
- Единый способ именования классов
- Легкая читаемость кода
- Повторное использование частей кода

Использование других типов селекторов при стилизации сведено к минимуму.

2.4.2 Адаптивность - отзывчивость сайта

Макет сайта в Figma выполнен в четырех версиях - для десктопа, двух вариантов планшета и смартфона для обеспечения адаптивности сайта при просмотре страниц сайта на этих устройствах.

Каждая из версий оптимизирована под свое устройство с применением следующих приемов:

- Изменение размеров элементов включая пропорции элементов относительно друг друга в рамках блока
- Меньшее количество столбцов на мобильных устройствах
- Перестроение расположения элементов внутри блока
- Сокращение размеров текста и межстрочных интервалов от десктопной к мобильной версии
- Изменение горизонтальных и вертикальных отступов
- Уменьшение размеров изображений

- Скрытие второстепенных изображений на мобильной версии
- Добавление новых элементов дизайна
- Перевод линейного навигационного меню в бургер меню в мобильной версии

Стоит отметить, что при стилизации html страницы используются такие приемы, которые не только позволяют реализовать задуманный дизайн макета уровня адаптивности на заданных устройствах, но в целом направлены на обеспечение максимальной отзывчивости страницы сайта на любых типах устройств, например:

- для позиционирования блоков / элементов используются гибкие элементы Flexbox и Grid layout
- Ширина и высота блочных элементов задается с использованием адаптивных свойств max/min-width и max/min-height там, где это целесообразно
- Используются относительные единицы для задания размеров элементов там, где применимо - %, vh, vw
- Центрирование блоков с применением padding

Для управления стилями элементов, на заданных макетом значениях ширины устройств настраивается метатег viewport и используются медиазапросы:

Чтобы подключить тег meta viewport в раздел head файла index.html добавляется строка:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Для адаптивного дизайна значения атрибута content viewport определяться двумя параметрами:

- width=device-width - ширина видимой области веб-страницы равняется CSS ширине устройства

- `initial-scale=1` - первоначальный масштаб веб-страницы равный 1 означает то, что масштаб равен 100%

Медиазапросы написаны для трех значений ширины экранов и срабатывают, когда ширина экрана становится меньше указанной в условии.

```
> @media (max-width:768px) { ...  
}  
> @media (max-width:700px) { ...  
}  
> @media (max-width: 460px) { ...  
}
```

В целом, все элементы html страницы стилизованы на основании данных указанных в шаблоне Figma:

- Параметры шрифта - название, размер, цвет, жирность
- Межстрочные интервалы
- Внешние и внутренние отступы элементов
- Цвет фона
- Свойства границ элементов – цвет, стиль, ширина, радиус
- Расстояние между элементами

Все эти параметры нужным образом изменяются в соответствии с дизайном макета подстраиваясь под характеристики устройства просмотра.

2.4.3 Использование изображений

В проекте применяются два типа изображений - растровые и векторные.

Растровые изображения загружены из макета Figma в формате jpeg для картинок прямоугольной формы, а векторные изображения в формате svg для иконок и непрямоугольных картинок.

После скачивания из макета файлов в формате jpeg их размер был оптимизирован в приложении TinyPNG, что позволило снизить их общий вес на 64% без потери качества изображения.



Для добавления изображений на страницу используются следующие способы:

- тег `` в html коде как для jpeg, так и для svg изображений
- тег `<svg>` в html коде
- через свойство `background-image` в CSS при использовании изображения в качестве фона

```

```

```
<svg class="header__burger-icon" xmlns="http://www.w3.org/2000/svg"
width="30" height="30" ...
</svg>
```

```
background-image: url(../../img/new-collection_background.jpg);
```

Изображения хранятся в папке `images` подключаются через относительные ссылки.

2.4.4 Загрузка и подключение шрифтов

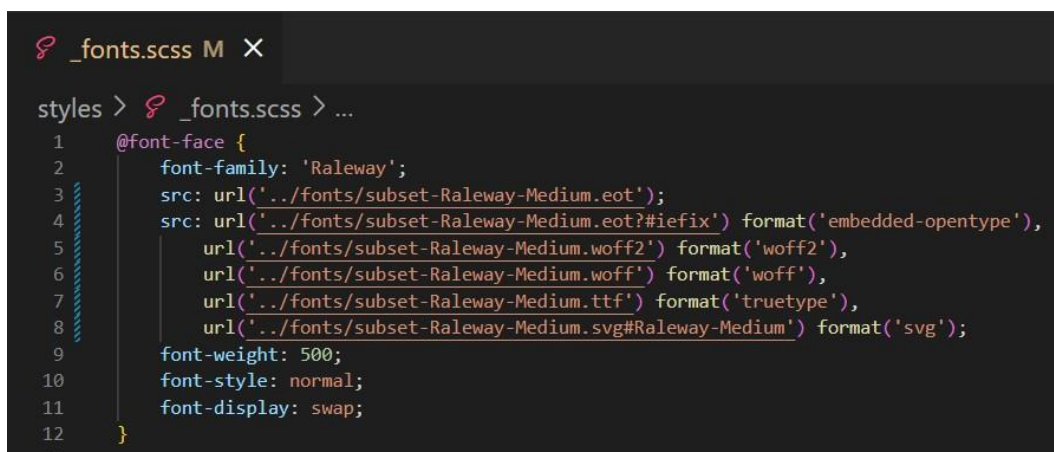
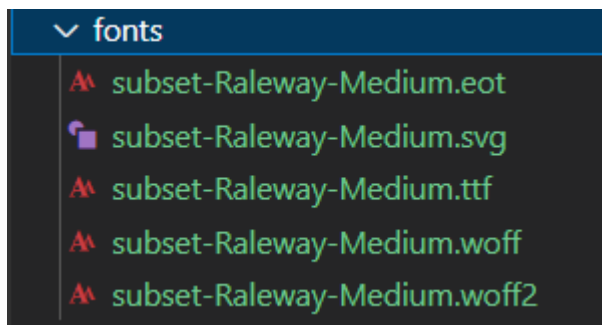


В макете Figma задан один нестандартный шрифт - `font-family: Raleway` со стилем `font-style: normal`;

Для подключения данного шрифта к проекту используется правило `@font-face` и генератор шрифтов приложения `transfonter.org`, который на основе загруженного

шаблона шрифта создает шрифты в нужных форматах и формирует `@font-face` правило для подключения к проекту.

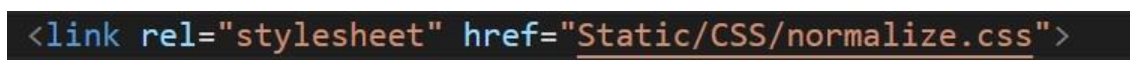
Сгенерированные в transfonter.org шрифты в форматах TTF, OTF, WOFF, WOFF2, SVG, для корректного отображения шрифта Raleway в разных браузерах, сохранены в папке `fonts`.



Правило `@font-face` прописано для всех указанных типов шрифтов в файле `_fonts.scss`, который подключается к основному Sass файлу - `index.scss` через директиву `@import 'fonts';`

2.4.5 Нормализация настроек

Для улучшения кроссбраузерности сайта подключен файл `normalize.css` в теге `<head>` файла `index.html`.



Проект `normalize.css` создан как альтернатива подходу со “сбросом исходных настроек” `CSS Reset`. Готовый для подключения файл с `CSS`-кодом можно скачать с сайта github.com.

Основные цели `normalize.css`:

- сохранение полезных настроек браузера, а не удаление их
- нормализация стилей для многих `HTML`-элементов
- корректировка ошибок и основные несоответствия браузеров
- повышение удобства использования;

`CSS Reset` охватывает все основные браузеры и нормализует `HTML5`-элементы, типографику, списки, встраиваемые элементы, формы и таблицы, при этом там где возможно сохраняются стандартные настройки.

Для контроля отображения сайта на различных параметрах экрана на этапе разработки использовался встроенный функционал отладчика браузера - эмуляция мобильных устройств.

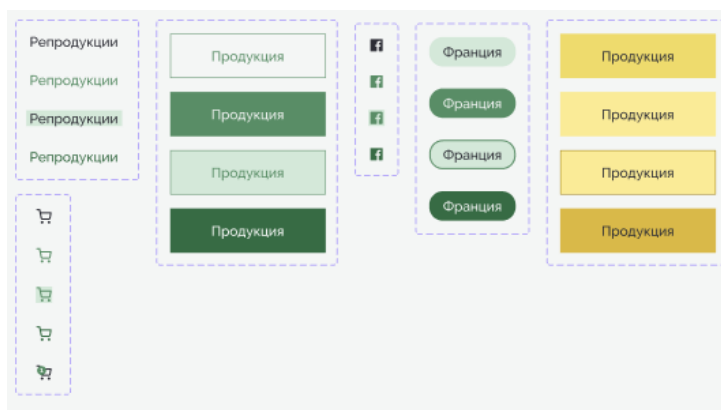
2.5 Создание интерактивных элементов главной страницы сайта

Интерактивность элементов сайта организована как за счет применения функционала HTML5, CSS3 так и JavaScript.

Примеры анимации, реализованные в проекте с использованием возможностей CSS3:

Эффекты наведения

Дизайном макета сайта в Figma заложено изменение внешнего вида кнопок и некоторых иконок для четырех состояний этих элементов: default, hover, focus, active.



Изменение стилей элементов при наведении мышью

Стандартные псевдоклассы `hover`, `focus`, `active` позволяют в полной мере выполнить задачу придания нужного стиля элементу при изменении его состояния, а возможность вложенности классов SASS придают хорошую читаемость коду.

```

&_button {
  @include button;
  padding: 20px 60px;
  background-color: $background_common;
  color: $button_color_1;
  border: 1px solid $button_color_1;

  &:hover {
    background-color: $button_color_1;
    color: $text_color_3;
    border: solid 1px transparent;
  }
  &:focus {
    background-color: $button_bgc_focus;
    color: $button_color_1;
    border: solid 1px $button_color_1;
  }
  &:active {
    background-color: $button_bgc_nav_active;
    color: $text_color_3;
    border: solid 1px transparent;
  }
}

```

Навигационное меню

Переходы по странице сайта при нажатии кнопок меню навигации реализованы через элемент html `<a>`, создающем гиперссылки на нужные места главной страницы через якорные id присвоенные целевым элементам.

```

<nav class="header_nav">
  <ul class="header_ul">
    <li class="header_li"><a href="#catalog" class="header_link">Репродукции</a> </li>
    <li class="header_li"><a href="#new" class="header_link">Новинки</a></li>
    <li class="header_li"><a href="#about" class="header_link">О нас</a></li>
  </ul>

```

Адаптивная - отзывчивая перестройка элементов сайта

Кроме приемов, указанных в п. Адаптивность - отзывчивость сайта текущей главы, стили ряда элементов переопределяются автоматически или при достижении значений, прописанных в медиа-запросах, например:

- Переключение значений свойства display
- Появление или скрытие элементов
- Изменение свойств выравнивая элементов внутри блоков
- Перестройка направлений выравнивания внутри контейнеров

Бургер меню

В мобильной версии макета сайта основное навигационное меню перестраивается из линейной компоновки в скрытую типа бургер меню.

С целью отработки различных приемов фронтенд разработки в проекте бургер меню реализовано в двух вариантах:

- 1) на HTML5/CSS3
- 2) JavaScript.

Вариант Бургер меню на HTML5/CSS3

В файле index.html создан блок header_burger в котором находится код относящейся к функционалу бургер меню. Сделано привязывание иконки с бургер меню в формате svg с чек-боксом располагая ее внутри тега label, связанного по id="burger_check" с тегом <input> типа "checkbox".

```
<!-- Burger menu on HTML5/CSS3 start-->
<div class="header__burger">
  <label class="label" for="burger__check"> <svg class="header__burger-icon" ...
  </svg> </label>
  <input class="header__check" id="burger__check" type="checkbox">
  <nav class="header__burger-nav">...
  </nav>
</div>
<div class="header__basket-box">...
</div>
<!-- Burger menu on HTML5/CSS3 end-->
```

Далее создан блочный элемент с навигационным меню, которое будет отображается на экране при изменении статуса checkbox на checked.

```
<nav class="header__burger-nav">
  <ul class="header__menu">
    <li class="header__li"><svg xmlns="http://www.w3.org/2000/svg" width="30" height="30" ...
    </svg></li>
    <li class="header__li"><a href="#catalog" class="header__link">Репродукции</a> </li>
    <li class="header__li"><a href="#new" class="header__link">Новинки</a></li>
    <li class="header__li"><a href="#about" class="header__link">О нас</a></li>
  </ul>
</nav>
```

Блоку навигационного меню `<nav class="header__burger-nav">` в файле `index.scss` задано абсолютное позиционирование и расстояния границ этого блока относительно его родительского элемента с относительным позиционированием, а также прописаны другие свойства для нужного позиционирования это блока.

```
&__burger-nav {
  position: absolute;
  width: 95vw;
  height: 100vh;
  background-color: orange;
  left: 0px;
  top: 30px;
  z-index: -1;
  visibility: hidden;
}
```

Применяется псевдокласс `:checked` для отображения/скрытия блока навигационного меню при изменении состояния `checkbox` и добавляется эффект плавного появления меню с левой стороны экрана используя возможности CSS3 анимации.

```
&:checked~.header__burger-nav {
  left: -23px;
  z-index: 1;
  -webkit-animation: slide-in-left 0.5s cubic-bezier(0.250, 0.460, 0.450, 0.940) both;
  animation: slide-in-left 0.5s cubic-bezier(0.250, 0.460, 0.450, 0.940) both;
  visibility: visible;
}
```

```
@keyframes slide-in-left {
  0% {
    -webkit-transform: translateX(-1000px);
    transform: translateX(-1000px);
    opacity: 0;
  }
  100% {
    -webkit-transform: translateX(0);
    transform: translateX(0);
    opacity: 1;
  }
}
```

Для создания анимации используется библиотека CSS анимации animista.net

animista

ON-DEMAND CSS ANIMATIONS LIBRARY

В заключении убирается значок чекбокса из зоны видимости экрана:

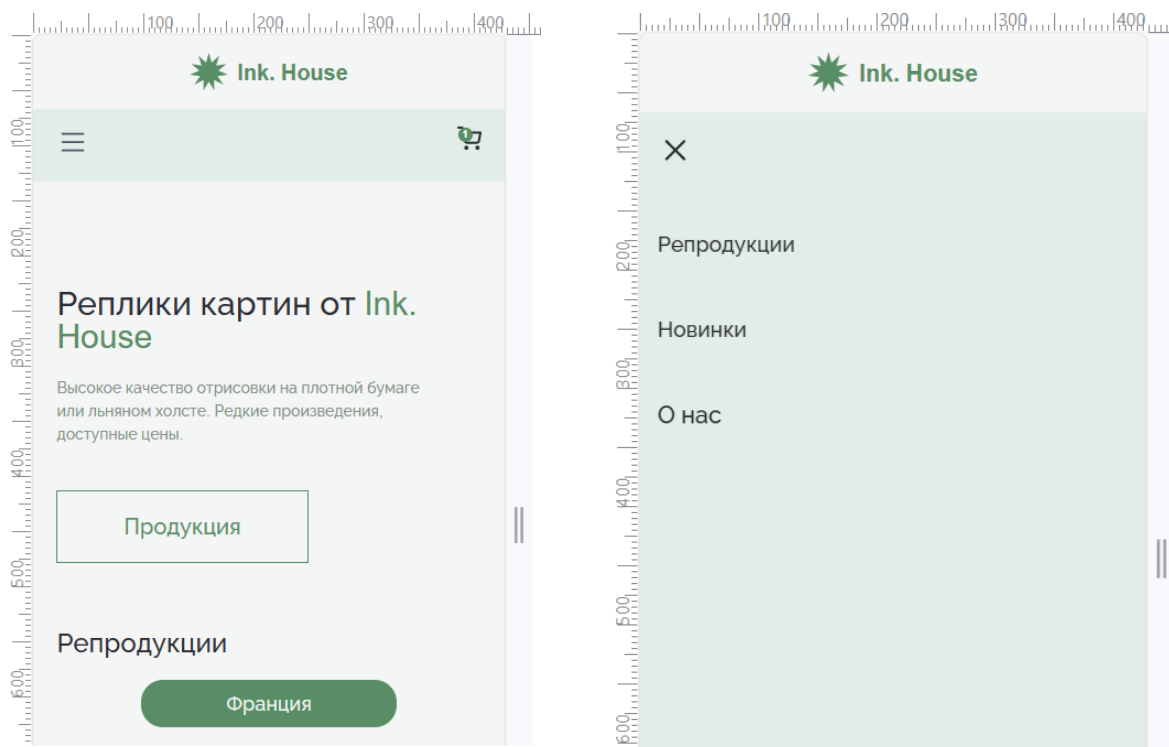
```
&__check {  
    position: absolute;  
    visibility: hidden;  
    left: -999999px;
```

Вариант Бургер меню на JavaScript

Возможности JavaScript позволяют создать более функциональное Бургер меню в точном соответствии с макетом сайта, сократить html и CSS код при этом реализация в JavaScript коде простая и понятная. Поэтому это решение используется в проекте в качестве основного.

Суть заключается использовании метода `addEventListener` для добавления обработчика события на html элементы бургер меню (символ бургер меню, символ закрытия меню и пункты меню) когда при нажатии левой кнопкой мыши на эти элементы происходит добавлении/ удалении CSS стилей приводящих к появлению меню на экране, закрытию меню, переходу на определенное положение на странице сайта в соответствии выбранном пунктом меню и закрытие меню.

Появление меню выполнено также, как и в Варианте 1) с использованием CSS3 анимации, а закрытие меню с другим анимационным эффектом через отдельный символ закрытия на навигационном меню.



Код JavaScript бургер меню находится в файле `burger.js` в папке `scripts` и подключается к файлу `index.html` в теге `head`.

```
<script src="scripts/burger.js" defer></script>
```

Файлы JavaScript относящиеся к решению других задач также располагаются в папке `script` и подключаются к файлу `index.html` аналогичным образом.

Переключение вкладок

Макет сайта содержит раздел Репродукции предусматривающий изменение отображаемых карточек с репродукциями картин при нажатии любую из трех кнопок с названием страны.



Структура блока с кнопками и карточками репродукций в html:

```
<div class="tab">
  <button class="tab_button tab--active">Франция</button>
  <button class="tab_button ">Германия</button>
  <button class="tab_button">Англия</button>
</div>
</div>
<div class="catalog_box">
  <div class="catalog_items content--active">...
</div>
  <div class="catalog_items ">...
</div>
  <div class="catalog_items">...
</div>
</div>
```

После верстки и стилизации этой части сайта, задача переключения вкладок реализуется на JavaScript в файле tabs.js.

Изначально активной является первая кнопка имеющая класс `tab--active` и блок с вкладками с классом `content--active`.

Переключение вкладок осуществляется за счет добавление класса `tab--active` и `content--active` на нажатую кнопку и соответствующую ей вкладку по событию “click” на любой из кнопок с классом `class="tab_button`.

С помощью метода `querySelectorAll()` по названию соответствующего класса получаем список всех кнопок и вкладок.

```
const tabEls = document.querySelectorAll(".tab__button");
const contentEls = document.querySelectorAll(".catalog__items");
```

Так как переключение вкладок осуществляется также из навигационного меню подвала сайта, то код относящийся к переключению вкладок оборачивается в функцию `function tabsSwitch(buttonNames)` для повторного использования при создании функционала переключения вкладок из нижнего меню. В качестве аргументов в эту функцию передаются название класса кнопок с которых будут переключаться вкладки.

Используя цикл добавляется обработчик событий `click` на все элементы списка `tabEls`.

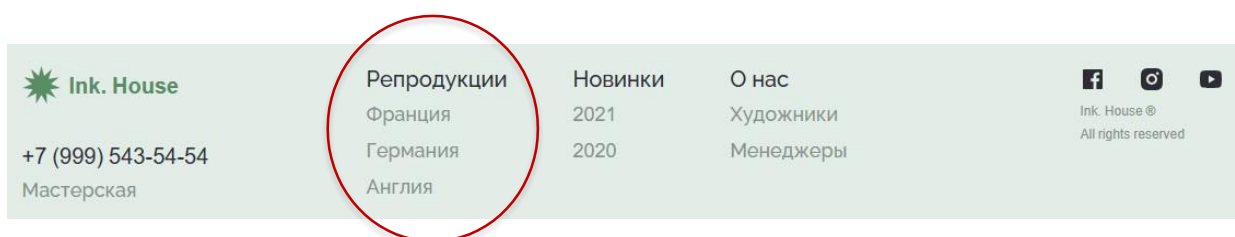
При наступлении события `click` производится удаление классов `tab--active` и `content--active` с использованием метода `forEach` со всех элементов списка кнопок и вкладок и затем добавление этих классов кнопке на которой сработало событие и соответствующей ей вкладке.

Предполагая, что действие по удалению и добавлению могут использоваться в других частях кода, для них написаны отдельные функции

```
function removeClassFromList(elName, ListName) и function addClassToListEl(elName, className) ,
```

которые вызываются внутри функции `tabsSwitch`.

Переходы и переключение кнопок вкладок из нижнего меню навигации.



Переход к разделу с репродукциями реализован с использованием функционала `html` по гиперссылке на `id` указанный в теге `<a>`.

Переключение кнопок вкладок и замена самих вкладок при нажатии на элементы навигационного меню, расположенного в подвале сайта создано на JavaScript по аналогии с вышеописанным переключением, более того для этого используется функция `function tabsSwitch(buttonNames)`, что позволяет избежать дублирование кода, делает программу компактной, легко читаемой и удобной для поддержки. С помощью метода `querySelectorAll()` по названию класса получаем список всех элементов меню `const footerTabEls`, которые используются при переключении. Далее вызывается функция `function tabsSwitch(buttonNames)` в которую передается созданный список элементов меню.

```
const footerTabEls = document.querySelectorAll(".footer__tab");  
tabsSwitch( buttonNames: footerTabEls);
```

Данный код расположен в файле tabs.js.

Счетчик товаров, добавленных в корзину и сохранения данных о товарах в localStorage

Программа для отображения количества товаров, добавленных в корзину и сохранения данных о них в хранилище браузера localStorage на главной странице сайта написана на JavaScript и находится в файле basketCounter.js.

Символ корзины расположенный в правом углу шапки сайта отображается без значка с количеством товаров, пока в корзину не добавлен хотя бы один товар.

При нажатии на кнопку “В корзину” карточки товара, на корзине появляется цифра с количеством товаров в корзине.



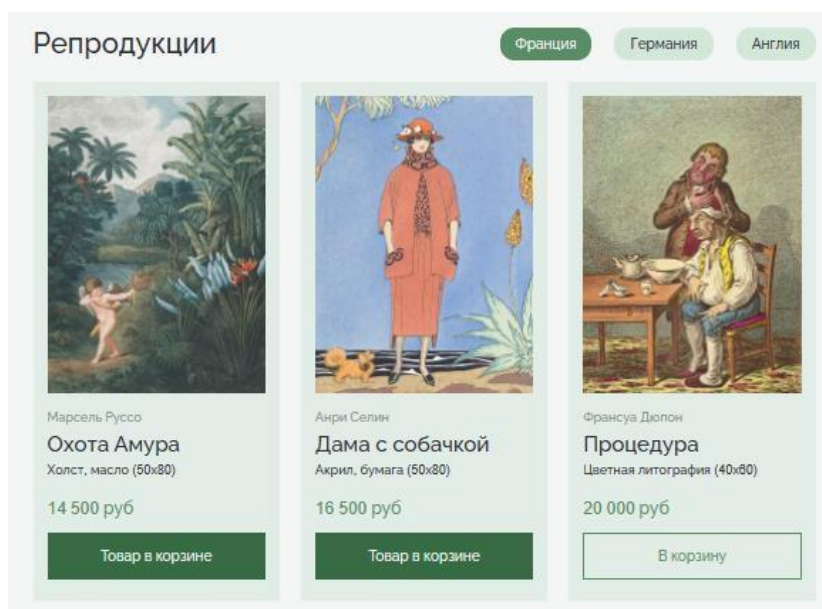
Выполняется добавление информации о выбранных товарах в localStorage следующим образом:

- Id номера товара указанный в дата-атрибуте data-number карточки товара добавляется в массив cartArray и далее данные из массива сохраняются в виде строки в localStorage используя метод `setItem(key, value)` и `JSON.stringify()`

- Начальное количество товаров равно одной штуки для всех выбранных товаров добавляется в массив `productItemQtyArray` и далее данные из массива сохраняются в виде строки в `localStorage` используя метод `setItem(key, value)` и `JSON.stringify()`
- Цена одного товара, указанная в дата-атрибуте `data-price` карточки товара добавляется в массив `productItemPriceArray` и далее данные из массива сохраняются в виде строки в `localStorage` используя метод `setItem(key, value)` и `JSON.stringify()`
- Значение счетчика числа товаров используя метод `setItem(key, value)`

Сохраненная в `localStorage` информация впоследствии используется на странице Корзина для реализации функционала этой веб-страницы.

Кроме этого клик кнопки “В корзину”, вызывает изменение текста на кнопке карточки товара с “В корзину” на “Товар в корзине”. В случае повторного нажатия на кнопку “Товар в корзине” увеличения числа товаров в корзине и записи данных в `localStorage` не происходит.



Этот функционал реализован следующим образом.

Используя метод `querySelector()` по названию класса находятся и добавляются элементы, которые будут использоваться в программе.

```
const headerBasketEl = document.querySelector(".header__basket");
const basketCounterEl = document.querySelector(".basket-counter");
const catalogBoxEl = document.querySelector(".catalog__box");
const cardButtonEl = document.querySelector(".card__button");
```

Добавляется событие “click” на элемент с каталогом карточек с помощью метода `addEventListener`.

В случае одновременного выполнения двух условий – нажатие происходит на элементе с классом `"card__button"` и текст внутри “В корзину” совершаются следующие действия:

- Счетчик товаров изначально равный нулю увеличивается на единицу
- Скрытый символ со счетчиком меняет свойство `visibility` с `hidden` на `visible`, что делает его видимым на странице сайта
- Меняется текст кнопки на “Товар в корзине”
- Меняется стилизация нажатой кнопки за счет удаления одного класса и добавлении другого

```
catalogBoxEl.addEventListener("click", (e) => {
  if (
    e.target.className === "card__button" &&
    e.target.innerText === "В корзину"
  ) {
    counter++;
    headerBasketEl.style.visibility = "visible";
    basketCounterEl.innerText = counter;
    e.target.innerText = "Товар в корзине";
    e.target.classList.remove("card__button");
    e.target.classList.add("catalog__card__button__basket");
  }
});
```

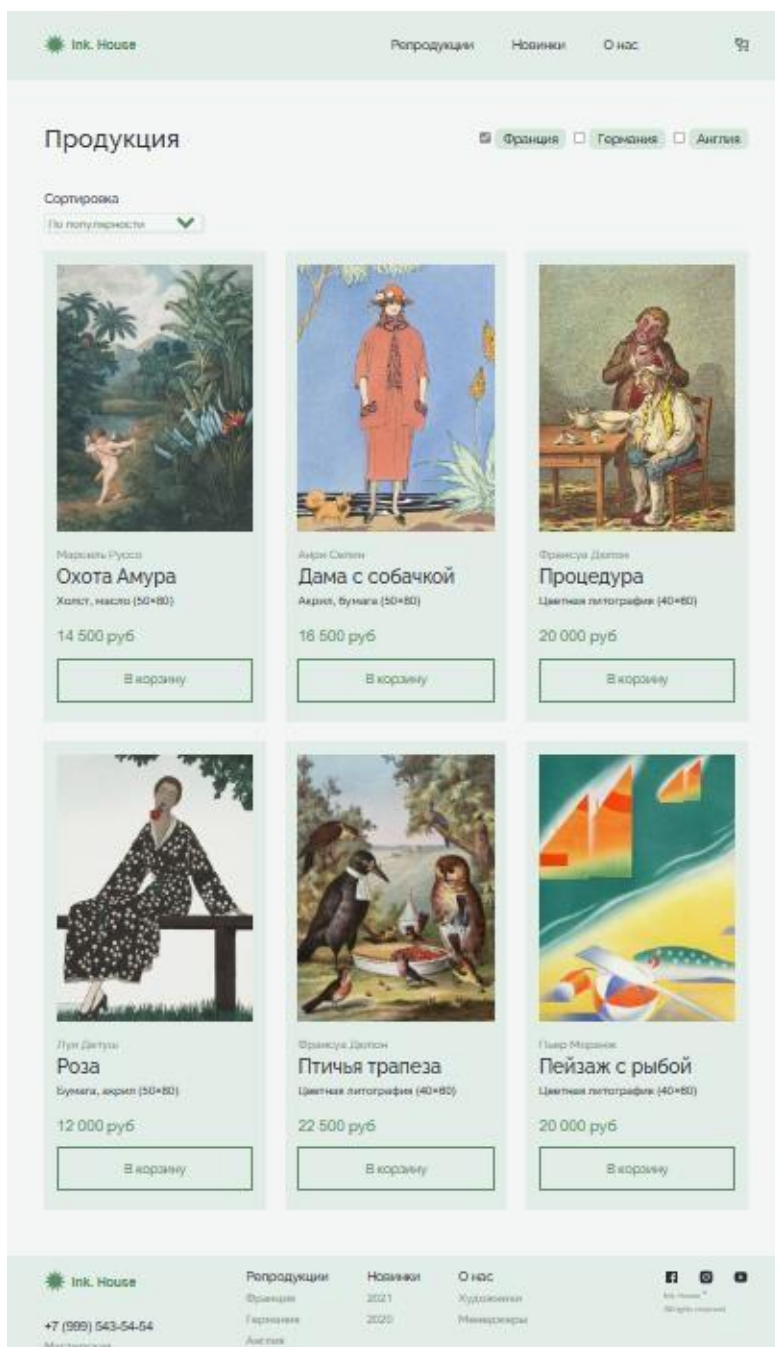

2.6 Разработка внутренних страниц сайта

Дополнительно к макету Figma были созданы четыре страницы сайта на основе собственного дизайна.

Верстка, стилизация и создание интерактивных элементов всех внутренних страниц сайта выполнена с использованием подходов аналогичных примененным при создании главной страницы сайта.

Ниже приводится обзор внутренних страниц сайта:

1. Страница Продукция - с возможностью поиска, фильтрации и добавлении товаров в корзину.



Скриншот общего вида страницы Продукция.

Переход с главной страницы сайта на данную страницу сайта осуществляется через кнопку `<button>` вложенную внутрь элемента `<form>` с атрибутом `action` содержащим адрес страницы.

Переход со страницы Продукция на главную страницу сайта может быть выполнен через навигационное меню шапки и подвала сайта.

HTML-код страницы находится в файле `products.html`. Стилизация выполнена с использованием возможностей CSS препроцессора SASS. Файлы со стилями хранятся в папке `styles`:

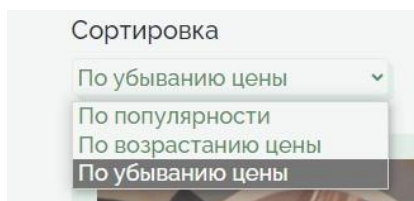
```
# products.css
# products.css.map
# products.scss
```

Интерактивный функционал реализован с помощью CSS3 и JavaScript. Файлы с кодом JavaScript хранятся в папке `scripts`: `products.js`, `basketCounter.js`, `burger.js`.

Интерактивный функционал на CSS3 – переходы на другие станицы сайта, эффекты наведения, меню сортировки, чек-бокс.

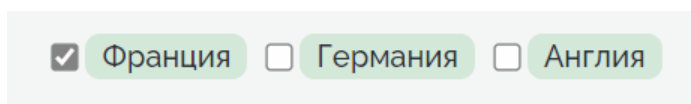
Интерактивный функционал на JavaScript:

- Сортировка товаров по нескольким критериям



По умолчанию на странице показываются товары, отсортированные по популярности.

- Фильтрация товаров по названию страны

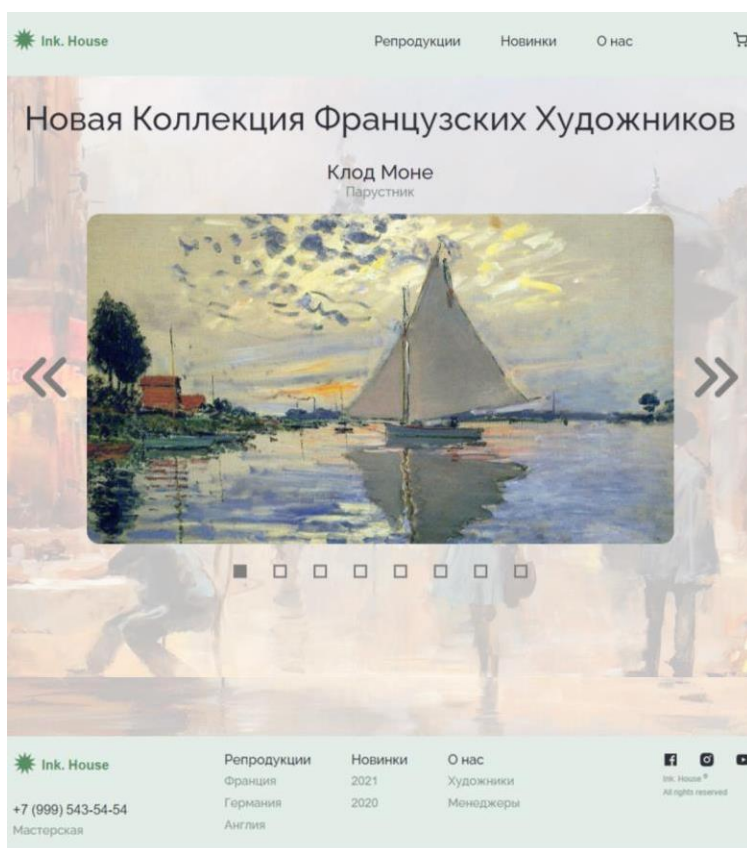


Изначально на странице показаны товары всех стран. При выборе страны или нескольких стран выводятся товары, принадлежащие этой стране/странам. Если все фильтры сняты на экране снова отображается все товары.

- Бургер меню – аналогично главной странице сайта
- Счетчик товаров, добавленных в корзину и сохранения данных о товарах в localStorage – аналогично главной странице сайта

При нажатии на кнопку - В корзину карточки товара, товар добавляется в корзину и счетчик товаров корзины увеличивается на единицу, а в хранилище браузера localStorage сохраняются данные с номерами карточек выбранных товаров и числом товаров.

2. Страница Новая коллекция – слайдер изображений картин французских художников



Скриншот общего вида страницы Новая коллекция.

Переход с главной страницы сайта на данную страницу сайта осуществляется через кнопку `<button>` вложенную внутрь элемента `<form>` с атрибутом `action` содержащим адрес страницы.

Переход со страницы Новая коллекция на главную страницу сайта может быть выполнен через навигационное меню шапки и подвала сайта.

HTML-код страницы находится в файле `frenchCollection.html`. Стилизация выполнена с использованием возможностей CSS препроцессора SASS. Файлы со стилями хранятся в папке `styles`:

```
# frenchCollection.css
# frenchCollection.css.map
# frenchCollection.scss
```

Интерактивный функционал реализован с помощью CSS3 и JavaScript. Файлы с кодом JavaScript хранятся в папке `scripts`: `slider.js`, `burger.js`, `basketCounterShort.js`.

Интерактивный функционал на CSS3 – переходы на другие станицы сайта, эффекты наведения.

Интерактивный функционал на JavaScript:

- Слайдер отображающий изображения картин новой коллекции французских художников

В слайдере реализована возможность переключения изображений как через кнопки Вперед-Назад, так и через навигационные точки (индикаторы) для быстрого переключения между изображениями.

При нажатии на кнопку Вперед  отображается следующее изображение, а

при нажатии на кнопку Назад  отображается предыдущее изображение.

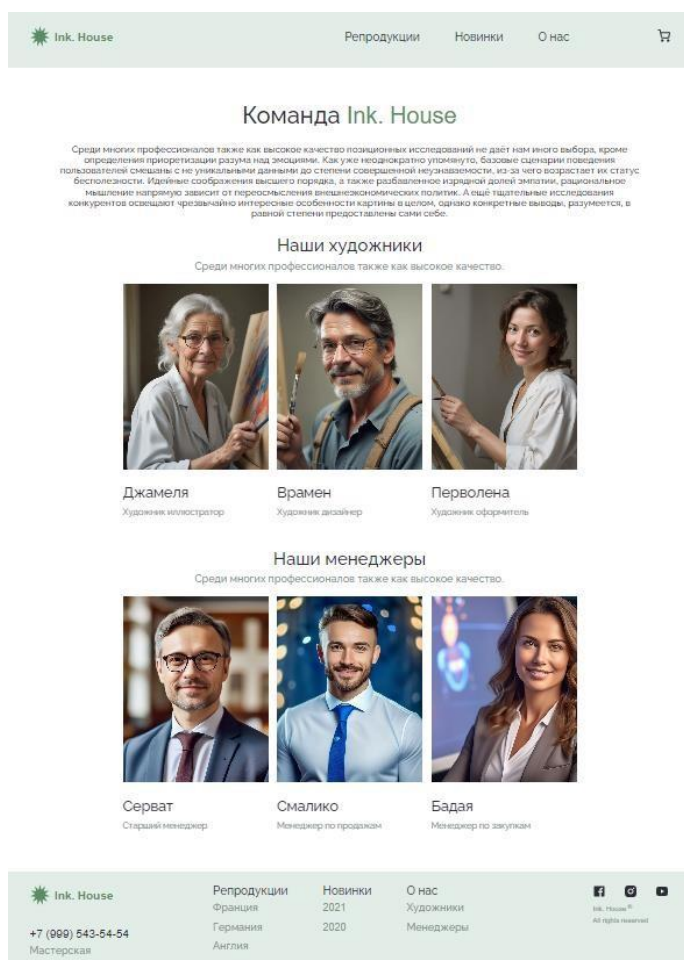
Переключение изображений осуществляется циклически, то есть после последнего изображения отображается первое, и наоборот.

При нажатии на навигационные точки, на экран выводится изображение, соответствующее нажатой точке, при этом меняется стиль этой точки.

- Счетчик товаров, добавленных в корзину

При загрузке страницы с помощью программы, находящейся в файле `basketCounterShort.js` используя метод `getItem(key)` из `localStorage` извлекается число товаров, добавленных в корзину и если полученное значение больше нуля, то на экран выводится это число над значком корзины.

3. Страница – Команда Ink. House – с возможностью просмотра подробной информации о сотруднике в модальном окне и различными эффектами анимации

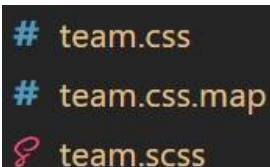


Скриншот общего вида страницы Команда Ink. House.

Переход с главной страницы сайта на данную страницу сайта осуществляется через гиперссылку тега `<a>`, в который обернут заголовок `Наша команда` раздела О нас.

Переход со страницы Команда Ink. House на главную страницу сайта может быть выполнен через навигационное меню шапки и подвала сайта.

HTML-код страницы находится в файле `team.html`. Стилизация выполнена с использованием возможностей CSS препроцессора SASS. Файлы со стилями хранятся в папке `styles`:



```
# team.css
# team.css.map
team.scss
```

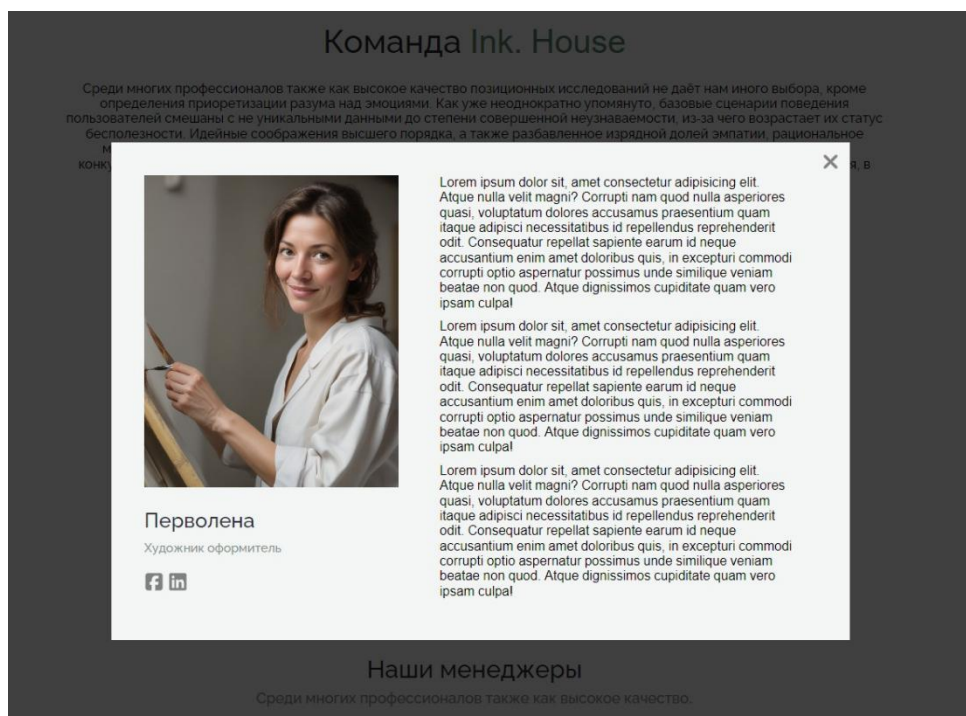
Интерактивный функционал реализован с помощью CSS3 и JavaScript. Файлы с кодом JavaScript хранятся в папке `scripts`: `team.js`, `burger.js` и `basketCounterShort.js`.

Интерактивный функционал на CSS3 – специальные эффекты появления элементов на экране при открытии страницы (фотографии сотрудников) и открытии модального окна, переходы на страницы социальных сетей, а также разнообразные эффекты наведения – например увеличение изображения фотографии сотрудника при наведении кнопки мыши.


Интерактивный функционал на JavaScript:

- Модальное окно с подробной информацией о выбранном сотруднике

При клике правой кнопкой мыши по фотографии сотрудника происходит затемнение фона экрана и открывается модальное окно с карточкой выбранного сотрудника.

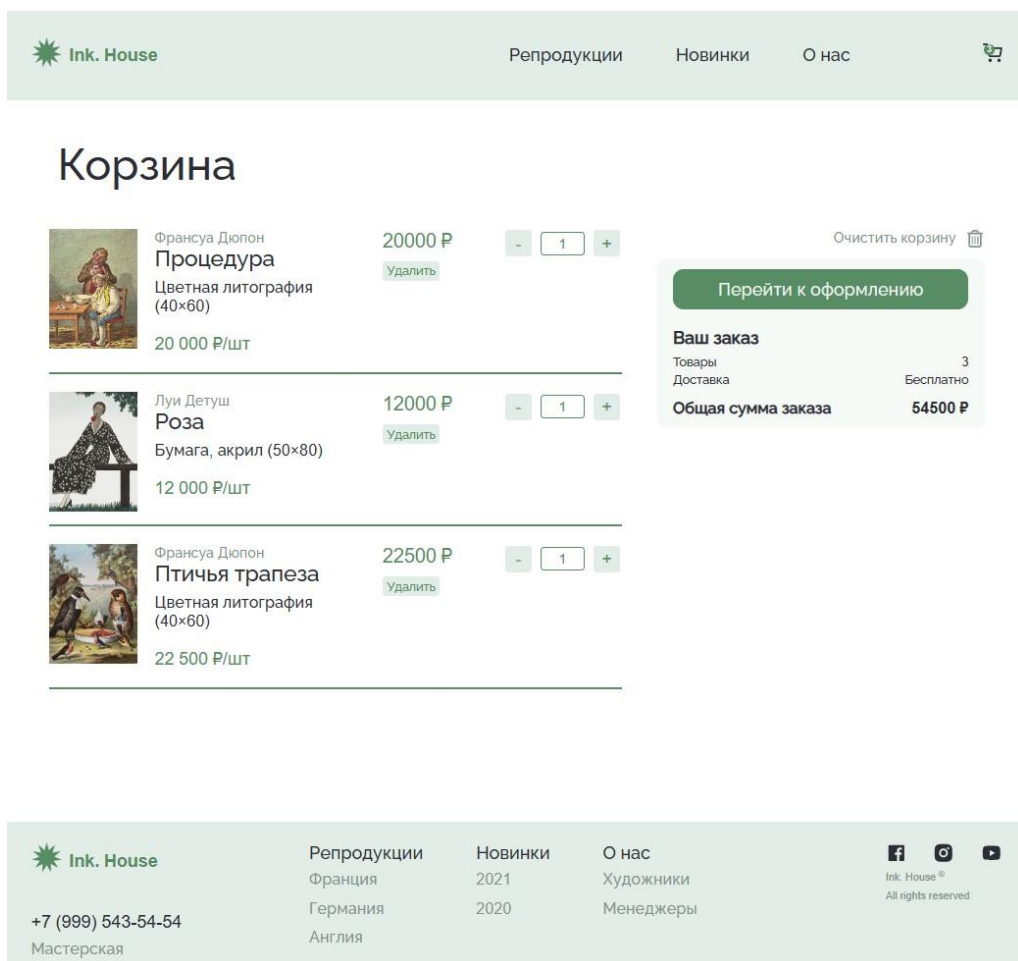


Через иконки социальных сетей можно перейти на станицу соответствующей социальной сети при клике мышью.

При нажатии на значок в правом верхнем углу  или любое место экрана кроме модального окна модальное окно закрывается.

- Бургер меню – аналогично главной станице сайта
- Счетчик товаров, добавленных в корзину – аналогично странице Новая коллекция

4. Страница Корзина – отображение карточек выбранных товаров, возможность изменения количества позиций товаров, удаление товара, подсчет общей цены товара и итоговой цены всех товаров в корзине



Скриншот общего вида страницы Корзина.

Переход на страницу Корзина может быть осуществлен с любой страницы сайта кликом правой кнопкой мыши по иконке корзины в шапке сайта. Для этого используется гиперссылка тега `<a>`, в которую обернута иконка корзины.

Переход со страницы Корзина на главную страницу сайта может быть выполнен через навигационное меню шапки и подвала сайта.

HTML-код страницы находится в файле `basket.html`. Стилизация выполнена с использованием возможностей CSS препроцессора SASS. Файлы со стилями хранятся в папке `styles`:

```
# basket.css
# basket.css.map
🔗 basket.scss
```

Интерактивный функционал реализован с помощью CSS3 и JavaScript. Файлы с кодом JavaScript хранятся в папке scripts: basket.js, burger.js и basketCounterShort.js.

Интерактивный функционал на CSS3 – специальные эффекты появления элементов на экране при открытии модального окна, переходы на другие страницы сайта, а также разнообразные эффекты наведения.

Интерактивный функционал на JavaScript:

- Отображение на экране карточек товаров, добавленных в корзину на основе данных сохранённых в localStorage - общее количество товаров, порядковые номера товаров, начальное количество товаров, цена товара
- Удаление товара с экрана и удаление из localStorage для данного товара его номера, цены, количества штук, пересчет и обновление общего количества товаров и общей суммы заказа в таблице Ваш Заказ, обновление общего количества товаров в localStorage при нажатии кнопки [Удалить](#)
- Увеличение или уменьшение единиц товара при нажатии кнопок + –. При этом происходит пересчет и обновление стоимости этого товара, в соответствии его количеством, общей стоимости заказа, обновляется количество единиц товара, а также новое число единиц товара и стоимости товара сохраняется в localStorage. Если количество единиц товара равно единице, то кнопка – неактивна [-](#)



Юджин Зиллион
Белый попугай
Цветная литография
(40×60)
15 500 Р/шт

48000 Р

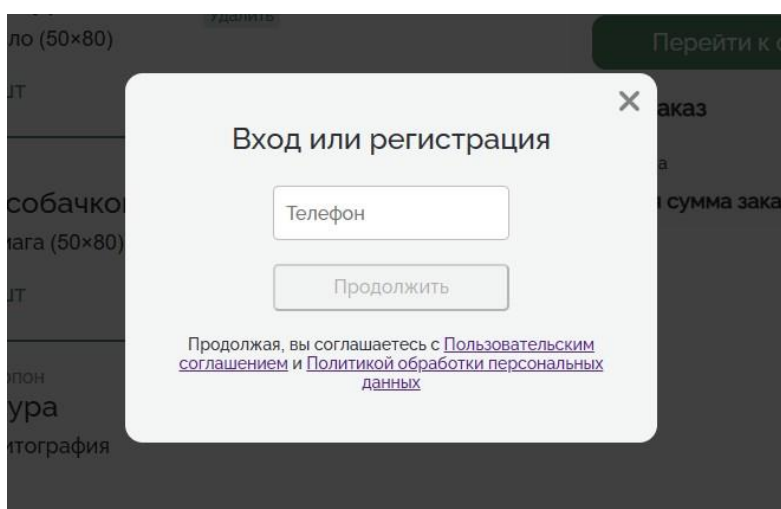
[Удалить](#)


[-](#) [+](#)

- Удаление всех товаров с экрана и всех данных о товаре из localStorage при нажатии на значок корзины 🗑
- Модальное окно с формой регистрации

Модальное окно реализовано аналогично модальному окну страницы Команда Ink. House.

При клике правой кнопкой мыши по кнопке Перейти к оформлению происходит затемнение фона экрана и открывается модальное окно с формой Авторизации/Регистрации.



При нажатии на значок в правом верхнем углу  или любое место экрана кроме модального окна модальное окно закрывается.

- Вывод на экран информационного сообщения, что корзина пуста, при клике на иконку корзины, когда в ней нет товаров

Корзина



В корзине пока пусто.

Выберите товары на [Главной странице](#) или на странице [Продукция](#).

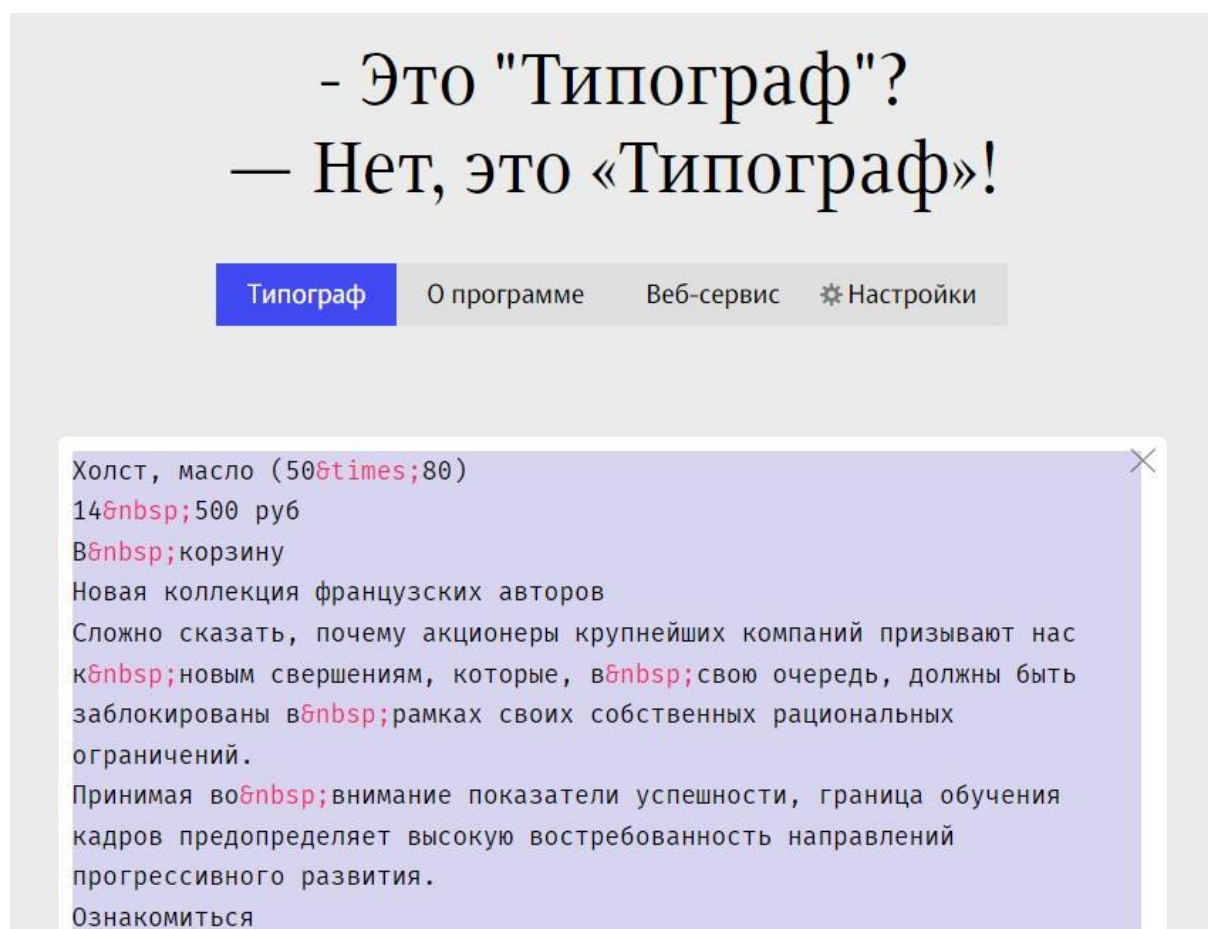
- Бургер меню – аналогично главной странице сайта
- Счетчик товаров, добавленных в корзину – аналогично странице Новая коллекция

2.7 Проверка веб-сайта на ошибки

Проверка синтаксиса и типографики текста

Проведена проверка текста на наличие синтаксических ошибок. Ошибок не обнаружено.

Для подготовки типографики текста сайта перед его загрузкой в интернет использована программа «Типограф» студии Лебедева - <https://www.artlebedev.ru/typograf/>



В соответствие с рекомендациями «Типограф» произведено редактирование текста.

Тестирование HTML кода

HTML код успешно проверен на валидность т.е. соответствие стандартам W3C (The World Wide Web Consortium) с использованием онлайн валидатора <https://validator.w3.org/>

Проверка на валидаторе охватывает следующие аспекты:

- Проверка синтаксических ошибок
- Проверка вложенности тегов
- Валидация DTD — анализ кода на соответствие Document Type Definition включающая проверку названий тегов, атрибутов, “встраивания” тегов
- Проверка на посторонние элементы, отсутствующие в DTD

Чистота HTML кода является важным фактором влияющим на скорость загрузки веб-страницы, корректность отображения содержимого сайта на разных устройствах и браузерах, качество поисковой индексации в поисковых системах.

The screenshot shows a web browser window with the address bar displaying the URL: `https://validator.w3.org/nu/?doc=https%3A%2F%2Fart-replica.vercel.app%2F`. The browser's taskbar at the bottom includes icons for Google, Downloads, Multitran, Gmail, YouTube, Maps, Translate, News, and other applications.

The main heading of the page is "Nu Html Checker". Below it, a note states: "This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change".

The results section is titled "Showing results for `https://art-replica.vercel.app/`". Under the "Checker Input" section, there are checkboxes for "Show" (source, outline, image report) and an "Options..." button. The "Check by" dropdown is set to "address", and the input field contains the URL `https://art-replica.vercel.app/`. A "Check" button is located below the input field.

A green banner at the bottom of the results section reads: "Document checking completed. No errors or warnings to show." Below this banner, it says: "Used the HTML parser. Externally specified character encoding was utf-8. Total execution time 25 milliseconds."

At the very bottom, there are links for "About this checker", "Report an issue", and the version number "Version: 24.2.27".

Функциональное тестирование

До и после размещения сайта в сети интернет проведено тестирование функционала сайта – меню навигации, работа кнопок, анимационные эффекты, переключение вкладок, счетчик товаров в корзине, корректность сохранения и удаления данных хранилища браузера `localStorage`, правильность расчетов и

реагирования на события на странице Корзина и других страницах сайта, открытие и закрытие модальных окон, переходы внутри и между страницами.

По результатам тестов были устранены обнаруженные недочеты в работе мобильной версии сайта.

Тестирование пользовательского интерфейса и опыта (UX/UI тестирование)

Проведено качественное UX/UI исследование на группе пользователей в форме краткого интервью по результатам тестового использования сайта на компьютере, планшете и смартфоне. Все участники опроса высоко оценили, как внешний вид страниц веб-сайта, так и удобство использования функционала сайта – меню, переходы, кнопки и т.д.

Тестирование совместимости с браузерами

Сайт был протестирован на всех широко используемых в России браузерах - Google Chrome, Yandex, Mozilla Firefox, Safari, Microsoft Edge, Opera.

Сайт продемонстрировал хорошую совместимость с этими популярными браузерами. Интерфейс и функциональность сайта работают должным образом вне зависимости от вида браузера на различных параметрах экрана.

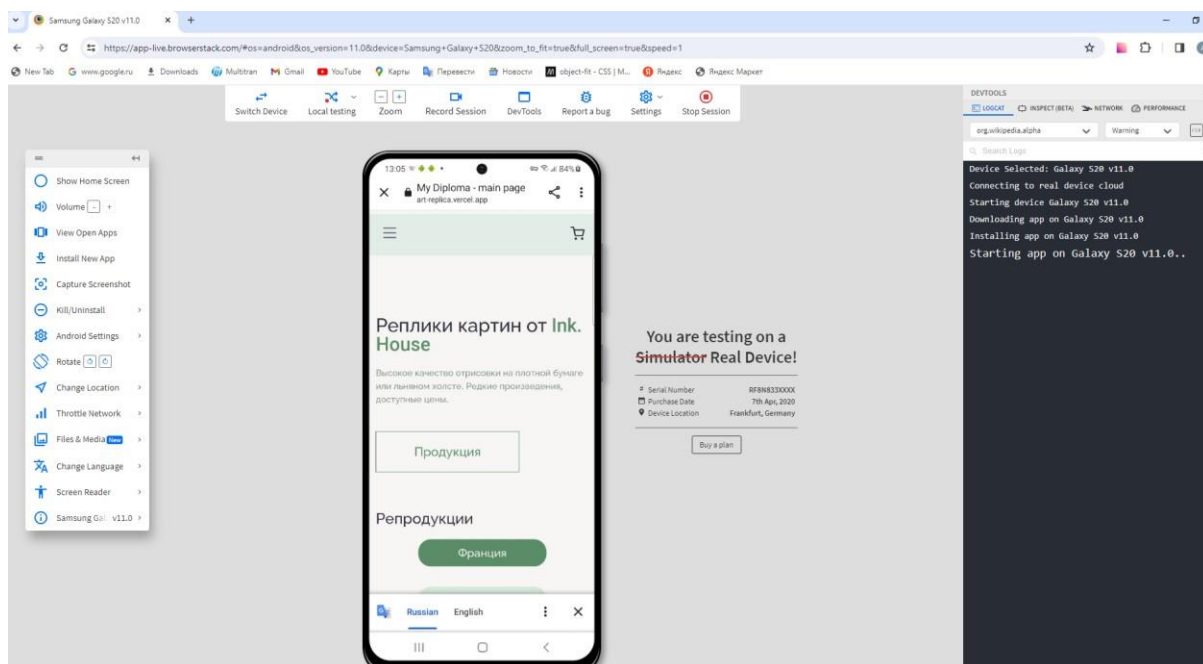
Тестирование совместимости с мобильными устройствами

Для тестирования работы сайта на мобильных устройствах использовался режим эмуляции мобильных устройств Chrome DevTools, удаленное подключение к мобильным устройствам через сервис Browserstack - <https://www.browserstack.com/live>, реальные смартфоны и планшеты.

Сайт был проверен на смартфонах и планшетах различных производителей с ОС Android и Apple iOS.

Проверка включала корректное отображение всех элементов интерфейса и работоспособность функционала сайта на всех доступных расширениях и положениях экрана.

Сайт корректно функционирует на всех видах проверяемых устройств, что подтверждает успешное выполнение одной из основных задач проекта по созданию адаптивного – отзывчивого веб-сайта.



Тестирование производительности и скорости загрузки

Тестирование производительности и скорости загрузки сайта проведено с использованием сервиса PageSpeed Insight - <https://pagespeed.web.dev/?hl=ru>

Результаты теста имеют высокие показатели всех основных характеристик, как на мобильных устройствах, так и на компьютере.

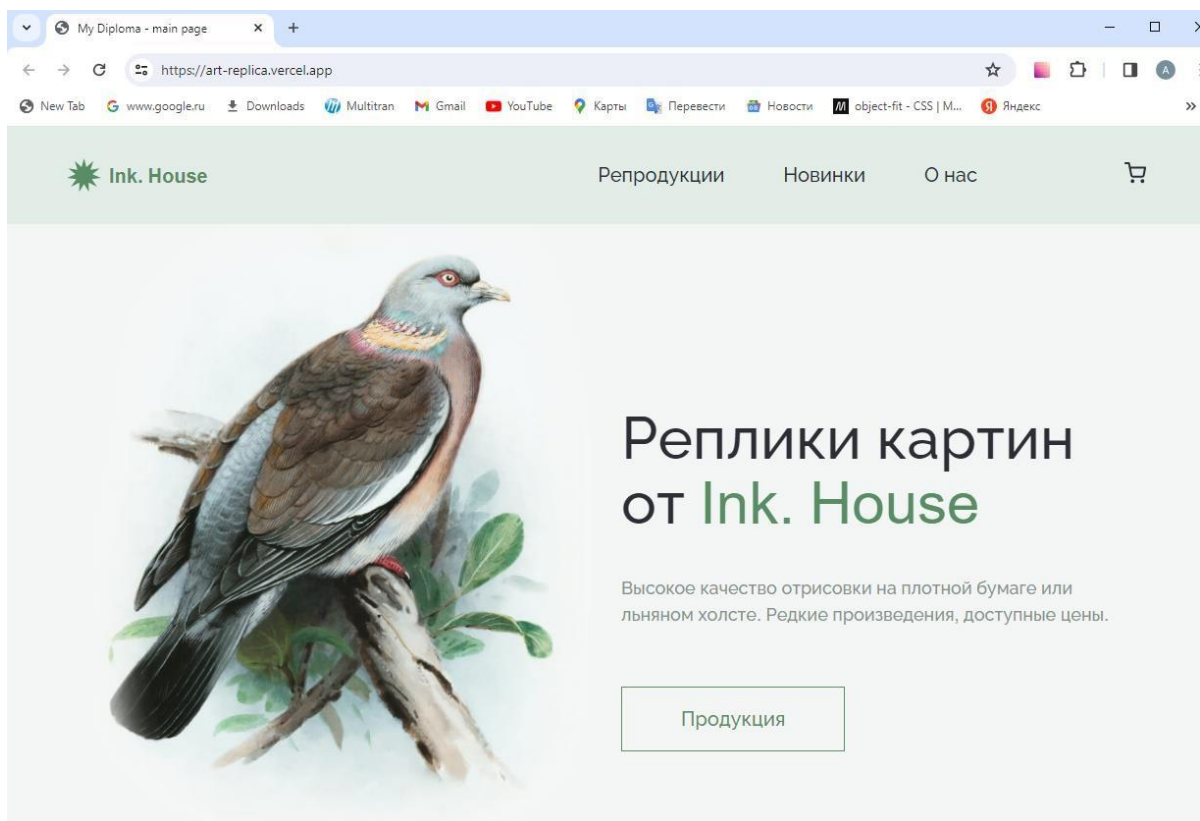
Стоит отметить высокую оценку скорости загрузки и показателя поисковой оптимизации, что является важными характеристиками для интернет магазинов так как позволяет обеспечить лучшую поисковую выдачу при запросах в сети и создает положительный пользовательский опыт благодаря быстрой работе сайта.

2.8 Загрузка веб-сайта в сеть интернет

Для размещения сайта в интернет использован сервис Vercel – <https://vercel.com>. Vercel является облачной платформой предоставляющей услуги по развертыванию и хостингу веб-приложений и сайтов.

Vercel позволяет осуществить деплоймент проекта напрямую из репозитория Git.

Адрес интернет сайта в сети интернет - <https://art-replica.vercel.app/>



ЗАКЛЮЧЕНИЕ

В ходе выполнения проекта удалось реализовать все планируемые задачи.

В Главе 1, которая в основном охватывает теоретические аспекты создания веб-сайтов проведен подробный анализ современных подходов к созданию веб-сайтов в результате чего определен способ и инструменты для реализации проекта.

Глава 2 описывает практическую сторону воплощения проекта в жизнь в ходе которой был выбран макет сайта, выполнена верстка, стилизация, добавление интерактивных элементов с использованием HTML5, CSS3, JavaScript и других инструментов веб-разработки.

Веб-сайт успешно прошел все основные виды тестов уместных для подобных проектов включая тестирование работы на различных типах устройств и кроссбраузерное тестирование.

Получены положительные отзывы от группы пользователей принявших участие в UX/UI тестировании сайта о качестве дизайна сайта и удобстве его использования.

Веб-сайт размещен в сети интернет - <https://art-replica.vercel.app/>

Все это объективно подтверждает соответствие веб-сайта современным требованиям фронтенд разработки, а также достижение основной цели проекта - создание адаптивного веб-сайта на основе макета Figma.

Более того первоначальный проект был значительно расширен за счет добавление внутренних страниц сайта разработанных по собственному дизайну, что сделало веб-сайт логически более завершенным и значительно расширило его функциональность.

Данный проект дал прекрасную возможность получить практический опыт по созданию веб-сайта, закрепить и расширить знания, полученные во время обучения в GeekBrains, определить направления для дальнейшего профессионального развития.

ЛИТЕРАТУРА

1. Бхаргава Адитья, А.Б. Грокаем алгоритмы. Руководство. Иллюстрированное пособие для программистов и любопытствующих / А.Б. Бхаргава Адитья. – Санкт-Петербург : Издательский Дом ПИТЕР, 2022. – 288 с. – ISBN 978-5-4461-0923-4
2. Бен, Фрейн HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Фрейн Бен. – Санкт-Петербург : Питер, 2018. – 304 с. – ISBN 978-5-496-00185-4.
3. Крокфорд Д. Как устроен JavaScript / Д. Крокфорд. – Санкт-Петербург : Питер, 2019. – 304 с.
4. Стефанов С. JavaScript. Шаблоны / С. Стефанов. – Санкт-Петербург : Символ-Плюс, 2011. – 250 с.
5. Симпсон, К. {Вы пока еще не знаете JS} Том 1-6 / К. Симпсон. – Санкт-Петербург : Питер.
6. Современный учебник JavaScript <https://learn.javascript.ru/>
7. Learn to Code - <https://www.w3schools.com/>
8. Дока — это документация для разработчиков <https://doka.guide/>
9. Отчет компании Global Digital 2023 Meltwater - <https://www.meltwater.com/en>
10. Отчеты компании GWI и Statista - <https://www.gwi.com/>
11. Отчеты компании GWI и Statista - <https://www.statista.com/>
12. Фронтенд-разработка: ключевые технологии и понятия - <https://habr.com/ru/companies/otus/articles/674748/>
13. Рейтинг CMS - <https://itrack.ru/rating-cms/>
14. Как работает Веб - Изучение веб-разработки | MDN - <https://developer.mozilla.org/>
15. Основы CSS - Изучение веб-разработки | MDN - <https://developer.mozilla.org/>
16. Публикация вашего веб-сайта - Изучение веб-разработки | MDN - <https://developer.mozilla.org/>

17. Методология БЭМ - <https://ru.bem.info/methodology/>,
<https://yandex.ru/dev/bem/>
18. SASS документация - <https://sass-lang.com/documentation/>, <https://sass-scss.ru/>
19. Web Application Architecture: How the Web Works -
<https://www.altexsoft.com/blog/web-application-architecture-how-the-web-works/>
20. Progressive Web Apps: Core Features, Architecture, Pros and Cons | AltexSoft -
<https://www.altexsoft.com/blog/progressive-web-apps/>
21. Курс Лекций Алевтины Шаталовой | Geekbrains Знакомство с веб-технологиями - <https://gb.ru/>
22. Курс лекций Игоря Зуриева | Geekbrains – Гибкие методологии - <https://gb.ru/>
23. Курс лекций Алексея Кадочникова | Geekbrains – Веб-вёрстка HTML/CSS, Продвинутый HTML + CSS, Основы Javascript, JavaScript про ECMAScript, JavaScript про API браузеров - <https://gb.ru/>

ПРИЛОЖЕНИЯ

1. Приложение №1

Презентация дипломного проекта на 7 страница — в файле -
Presentation.pdf



Presentation.pdf

2. Приложение №2

Код проекта — <https://github.com/Als-Kub/diploma-project>