

Geekbrains

Разработка фронтенд части вебсайта-портфолио с использованием фреймворка React.

|  |  |
| --- | --- |
|  | Программа: |
|  | Разработчик — Frontend разработка (React). |
|  | Акчалов Анвар Мухаметжанович |

Магнитогорск

2024

Содержание

[ВВЕДЕНИЕ 4](#_Toc183384711)

[1 ИССЛЕДОВАНИЕ И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc183384712)

[1.1 Современные методы веб-разработки 6](#_Toc183384713)

[1.2 Обзор существующих сайтов-портфолио и выявление лучших практик 8](#_Toc183384714)

[1.3 Планирование структуры и дизайна сайта-портфолио. 10](#_Toc183384715)

[1.3.1 Выбор и анализ макета Figma. 10](#_Toc183384716)

[1.4 Исследование предметной области – инструментов веб-разработки, технологий и библиотек. 14](#_Toc183384717)

[1.4.1 Pixso 14](#_Toc183384718)

[1.4.2 Visual Studio Code 16](#_Toc183384719)

[1.4.3 Git 19](#_Toc183384720)

[1.4.4 Vite 23](#_Toc183384721)

[1.4.5 HTML 24](#_Toc183384722)

[1.4.6 CSS 24](#_Toc183384723)

[1.4.7 SASS 30](#_Toc183384724)

[1.4.8 JavaScript 31](#_Toc183384725)

[1.4.9 React 32](#_Toc183384726)

[1.4.10 Redux 40](#_Toc183384727)

[1.4.11 Веб-браузеры и браузерные инструменты разработчика 44](#_Toc183384728)

[1.5 Вывод по главе 46](#_Toc183384729)

[2 РАЗРАБОТКА ФРОНТЕНД ЧАСТИ САЙТА 47](#_Toc183384730)

[2.1 Создание репозитория 47](#_Toc183384731)

[2.2 Развёртывание проекта 47](#_Toc183384732)

[2.3 Архитектура проекта. 48](#_Toc183384733)

[2.4 Создание компонентов JSX 49](#_Toc183384734)

[2.4.1 Компонент Container.jsx 49](#_Toc183384735)

[2.4.2 Компонент LeftMenu.jsx 50](#_Toc183384736)

[2.4.3 Компонент MainBlock.jsx 53](#_Toc183384737)

[2.4.4 Компонент RightMenu.jsx. 56](#_Toc183384738)

[2.5 Краткое описание проекта магазина одежды – Web Clothing Store 56](#_Toc183384739)

[2.6 Сборка и размещение сайта-портфолио на GitHub Pages 56](#_Toc183384740)

[2.7 Аудит проекта с использованием Lighthouse 58](#_Toc183384741)

[ЗАКЛЮЧЕНИЕ 60](#_Toc183384742)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 62](#_Toc183384743)

[Приложение А 64](#_Toc183384744)

ВВЕДЕНИЕ

Целью данного дипломного проекта является разработка фронтенд части вебсайта-портфолио с использованием фреймворка React. Вебсайт будет служить платформой для демонстрации навыков и опыта разработчика.

Фронтенд-разработка является важной и неотъемлемой частью проектирования вебсайтов. Данное направление активно развивается, предлагая множество инструментов и технологий для решения задач различного уровня сложности. Поэтому фронтенд-разработчикам важно использовать современный подход, отвечающий широкому перечню требовании, таких как:

* Отзывчивый дизайн;
* Быстрая загрузка;
* Доступность;
* Безопасность;
* Оптимизация для поисковых систем (SEO);
* Привлекательный и интуитивно понятный дизайн;
* Использование современных технологий;
* Знание принципов дизайна пользовательского интерфейса (UI) и взаимодействия с пользователем (UX);
* Соблюдение веб-стандартов;
* Тестирование и обеспечение качества.

Применение фреймворка React, позволяет решить большую часть данных задач.

Задачи:

1. Обзор существующих сайтов-портфолио и выявление лучших практик;
2. Планирование структуры и дизайна сайта-портфолио;
3. Исследование предметной области – фреймворка React, соответствующих библиотек и технологий;
4. Разработка фронтенд части сайта с использованием React и сопутствующих технологий;
5. Добавление и описание готовых проектов;
6. Тестирование и отладка сайта используя браузерные инструменты разработчика;
7. Развертывание сайта-портфолио на хостинге.

Для выполнения дипломного проекта будут использоваться следующие инструменты и технологии: Pixso, Visual Studio Code, Vite, Opera, Chrome, инструменты отладки, Git, GitHub, Beget, HTML/CSS, SASS (SCSS), JavaScript – React, Redux.

1. ИССЛЕДОВАНИЕ И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ
   1. Современные методы веб-разработки

Существует большое количество способов создания веб-платформ, но основа для любого сайта — это разработка фронтенд (клиентская) и бэкенд (серверная) частей.

Рассмотрим современные технологии для разработки frontend части. Самым простым способом является – API основанное на HTML, CSS, JavaScript. К более современным можно отнести фреймворки, такие как: ASP.NET, Ruby on Rails, Angular, Vue.js, React.js. Продвинутые способы основаны на C#/Java/JavaScript и включают в себя улучшенный функционал, готовые решения.

Для стилизации используют Bootstrap, Semantic UI, Foundation, Pure By Yahoo, Uikit, Material Design. Они так же включают в себя готовые решения и продвинутый функционал. А использование CMS – готовых шаблонов, основанных на WordPress, Drupal, Joomla или сайтов конструкторов Wix, Weebly, uKit, значительно ускоряет разработку сайта.

В настоящее время большую популярность имеют фреймворки, они позволяют структурированно создать крупномасштабные проекты, с возможностью дальнейшего масштабирования.

Для написания кода небольших сайтов используют текстовые редакторы, такие как Sublime, NotePad++, Brackets, но для больших проектов нужны серьезные инструменты в лице различных IDE, которые могут быть ориентированы под конкретную задачу.

IDE – среда разработки, имеет большое количество функций, которые облегчают написание, тестирование, отладку, сборку проекта.

Популярные IDE: Visual Studio Code, WebStorm, PyCharm, PhpStorm, Eclipse, Atom.

.

* 1. Обзор существующих сайтов-портфолио и выявление лучших практик

В ходе проведения всестороннего анализа существующих сайтов-портфолио, было изучено около десятка примеров с различных веб-ресурсов, проанализированы рекомендации популярных изданий и блогов для разработчиков. На основании анализа были выделены следующие ключевые характеристики:

* Цель портфолио;
* Четкая и лаконичная структура;
* Информация на сайте должна быть организована логично и легко доступна;
* Обратить внимание на то, что пользователи должны увидеть первым;

Например, выделить конкретные навыки в отдельный раздел, либо сделать так, чтобы блок с проектами появлялся раньше блока «Обо мне».

* Стараться придерживаться принципа «меньше, значит больше» ;

Это особенно важно учитывать при добавлении анимации. Наличие большого количества интерактивных и мигающих элементов на странице может быть избыточным, таким образом перегружая интерфейс с точки зрения UX/UI-дизайна. Необходимо обдумать, добавляют ли дополнительные элементы и анимация на странице какую-либо ценность.

* Проценты и шкалы навыков ни о чем не говорят;

Описание навыков с помощью процентов или индикаторов прогресса не добавляет информативности. Никто не знает, что это на самом деле означает. Если Вы скажите, что у Вас «100% навыков работы с Javascript, создастся впечатление, что Вы лучший разработчик Javascript в мире. Поэтому лучше использовать общепринятые и понятные показатели.

* Сократить количество кликов;

Простое, но эффективное правило UX, о котором следует помнить. Каждый клик на сайте должен иметь смысл.

* Сделать так, чтобы с автором страницы было легко связаться;

То есть необходимо разместить ссылки на электронную почту или социальные сети как в верхней, так и нижней части сайта.

* Призыв к действию;

На сайте есть четкий призыв к действию, например, связаться с разработчиком или нанять его для проекта

* 1. Планирование структуры и дизайна сайта-портфолио.

Так как помимо фронтенд-разработчика, я выступаю и дизайнером проекта, но при этом имею минимальный опыт работы c графическими программами, было принято решение выбрать готовый макет на онлайн-платформе сообщества Figma.

Сообщество Figma – это проект объединяющий, дизайнеров, разработчиков и других специалистов, использующих Figma.

Пользователи могут публиковать свои проекты Figma и делиться ими с другими членами сообщества. Это позволяет дизайнерам учиться друг у друга, находить вдохновение и переиспользовать существующие элементы в своих собственных проектах.

Сообщество Figma предоставляет форум, где пользователи могут задавать вопросы, получать помощь от других пользователей и общаться с командой Figma. Это ценный ресурс для начинающих дизайнеров и тех, кто сталкивается с проблемами при использовании Figma.

Макет должен отвечать параметрам, описанным в разделе «Обзор существующих сайтов-портфолио и выявление лучших практик».

* + 1. Выбор и анализ макета Figma.

Макет Figma выбранный для проекта располагается по ссылке - <https://www.figma.com/community/file/966954657029620690>.

Дизайн макета современный и минималистичный. Он использует нейтральную цветовую палитру с цветовыми акцентами жёлтого цвета.

Макет разделён на три основных вертикальных блока Left menu, Main menu, Right menu.

Left menu, Right menu имеют фиксированное положение, слева и справа соответственно. Main menu заполняется блоками сверху вниз – подразумевается прокрутка раздела.

Left menu включает следующие разделы:

* Карточка автора проекта: аватар, иконка активности, ФИО, профессия, иконки социальных сетей, возраст, резидент (резидентом какой страны является человек), фрилас (активен или нет), адрес;
* Блок со списком языков;
* Блок со списком навыков;
* Блок со списком дополнительных навыков;
* Кнопка для загрузки резюме.

Right menu включает следующие элементы:

* Иконка contrast, нажатие которой меняет цветовую тему;
* Иконки навигационного меню home, service, cv, portfolio, blog, contact.

При наведении на иконку навигации, над ней должен появиться чёрный блок с описанием раздела к которому относится ссылка.

Main menu включает следующие разделы:

* Блок с описанием информации о разработчике: ФИО, профессия, информация о себе, направляющая на раздел контактов кнопка «HIRE ME», большое изображение – фото разработчика, 7 цветных элементов в виде кружков, квадратов и треугольника, украшающих блок;
* Блок с описанием услуг, которые может предложить разработчик: заголовок «My Services», краткое описание предоставляемых услуг, карточка услуги (включает иконку), название услуги и её описание (если услуг много, то они формируются в таблицу из трёх столбцов);
* Блок с описанием стоимости услуг: заголовок «Price Plans», краткое описание тарифных планов, карточки тарифных планов с указанием названия плана, его стоимости и перечнем с указанием услуг, которые можно подключить;
* Блок с рекомендациями и отзывами о разработчике: заголовок «Recommendations», краткое описание раздела, карточки с отзывами (если отзывов больше трёх, то в нижней части раздела добавляются «точки» навигации, с возможностью прокрутки отзывов);
* Блок об образовании разработчика: заголовок «Education», краткое описание раздела, таблица (в правой части таблицы, указано место образования, дата, статус обучающегося, в левой части расположена информация о сертификатах об обучении);
* Блок о работе: «Work History» выполнен аналогично блоку об образовании с описанием занимаемой должности и месте работы;
* Блок портфолио: заголовок «Portfolio», описание раздела, фильтр по проектам (различаются типы проектов) в виде ссылок, сетка из квадратных карточек (в карточку вставлено изображение проекта, при клике на карточку подразумевается переход на проект);
* Блок с блогом: заголовок «Blog», описание раздела, карточки статей с изображением, названием, кратким описанием и ссылкой для перехода на статью;
* Блок с контактной информацией: в левой части расположена информация о посетителе – заголовок «Leave Us Your Info», ФИО, e-mail, предметная область сообщения, текст сообщения, кнопка «SEND MESSAGE»; в правой части – три вертикально расположенных блока с адресом, контактами и телефоном, в шапке блока заголовок «Contact Information»;
* Блок с интерактивной картой местоположения, либо статичное изображение с точкой адреса;
* Блок с логотипами;
* Футер: заголовок «2021 All Rights Reserved. Ojjomedia».

Так же для проекта представлены образцы иконок, кнопок, предоставлена цветовая схема.

Учитывая период времени, отведённый на выполнение итогового задания, изученные ранее инструменты веб-разработки и опыт работы в данной сфере, была принята следующая структура сайта-портфолио:

* Блок Left menu без аватара;
* Блок Right menu без иконки contrast;
* Блок Main menu без разделов «Price Plans», «Recommendations», «Work History», «Blog», карты и логотипов.

Пример макета представлен в приложении А.

* 1. Исследование предметной области – инструментов веб-разработки, технологий и библиотек.

1.4.1 Pixso

Pixso — это облачный инструмент для дизайна и прототипирования, который предлагает возможности, аналогичные Figma (рисунок 1.1). Он создан с учетом потребностей UX/UI-дизайнеров и позволяет работать над проектами в режиме реального времени, предоставляя функции для совместной работы, управления компонентами и создания интерактивных прототипов. Pixso позиционируется как удобная альтернатива Figma, которая обеспечивает похожий опыт взаимодействия с возможностью эффективного командного дизайна.

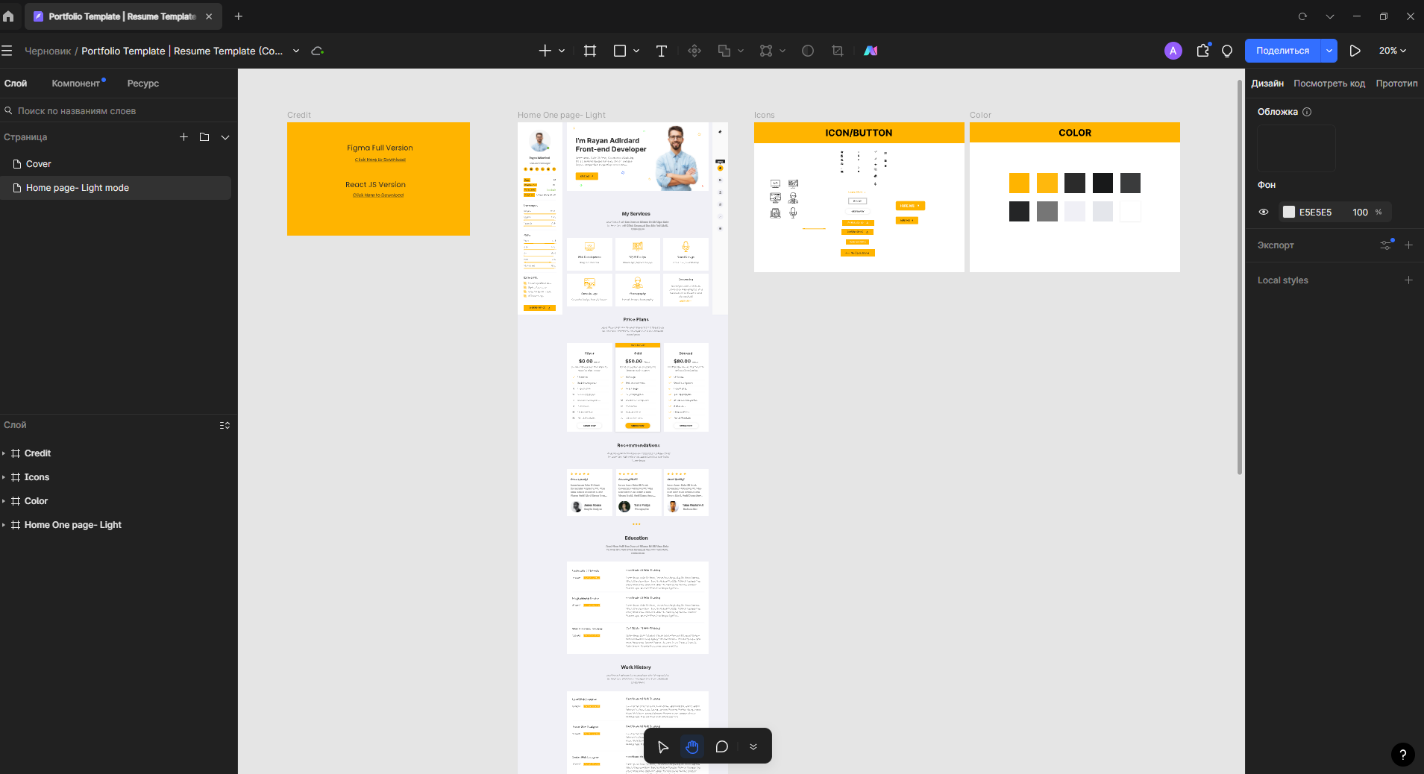


Рисунок 1.1 – Интерфейс Pixso

Pixso предлагает более гибкую ценовую политику, что делает его привлекательным для небольших команд и фрилансеров. Учитывая, что стоимость подписки Figma увеличивается по мере расширения команды, Pixso может стать более экономичным решением. Кроме того, Pixso предлагает бесплатный план с достаточным набором функций, что позволяет начать работать без дополнительных затрат.

Pixso разработан таким образом, чтобы напоминать интерфейс Figma, что позволяет дизайнерам легко адаптироваться к новому инструменту. Пользователи, знакомые с Figma, смогут быстро освоиться в Pixso, поскольку многие элементы и функции схожи, что упрощает переход на новую платформу и не требует длительного обучения.

Как и Figma, Pixso поддерживает режимы командной работы в реальном времени. Пользователи могут оставлять комментарии, предлагать правки, совместно разрабатывать дизайн и отслеживать изменения в проекте. Поддержка совместного редактирования позволяет командам эффективно взаимодействовать, даже если участники находятся в разных местах.

Pixso поддерживает работу с компонентами и библиотеками, что позволяет создавать дизайн-системы и управлять стилями в рамках одного проекта. Это особенно полезно для больших команд, где важно поддерживать единый стиль. Возможности Pixso по созданию компонентов позволяют организовывать дизайн в единый, структурированный и управляемый формат.

Pixso предлагает инструменты для создания интерактивных прототипов, позволяя дизайнерам тестировать пользовательские сценарии и демонстрировать функциональные возможности продукта. Это облегчает процесс создания реалистичных прототипов и помогает быстрее проводить тестирование.

В процессе обучения в Geekbrains для работы с макетами предлагалось использовать такие инструменты как Figma, Marsy, однако, учитывая, что на бесплатном плане Pixso имеет весь необходимый функционал, выбор был сделан в пользу этого инструмента.

1.4.2 Visual Studio Code

Visual Studio Code (VSCode) является популярным редактором кода для разработки программного обеспечения. Он предоставляет широкий спектр функций, которые делают его идеальным для веб-разработки (рисунок 2.1).

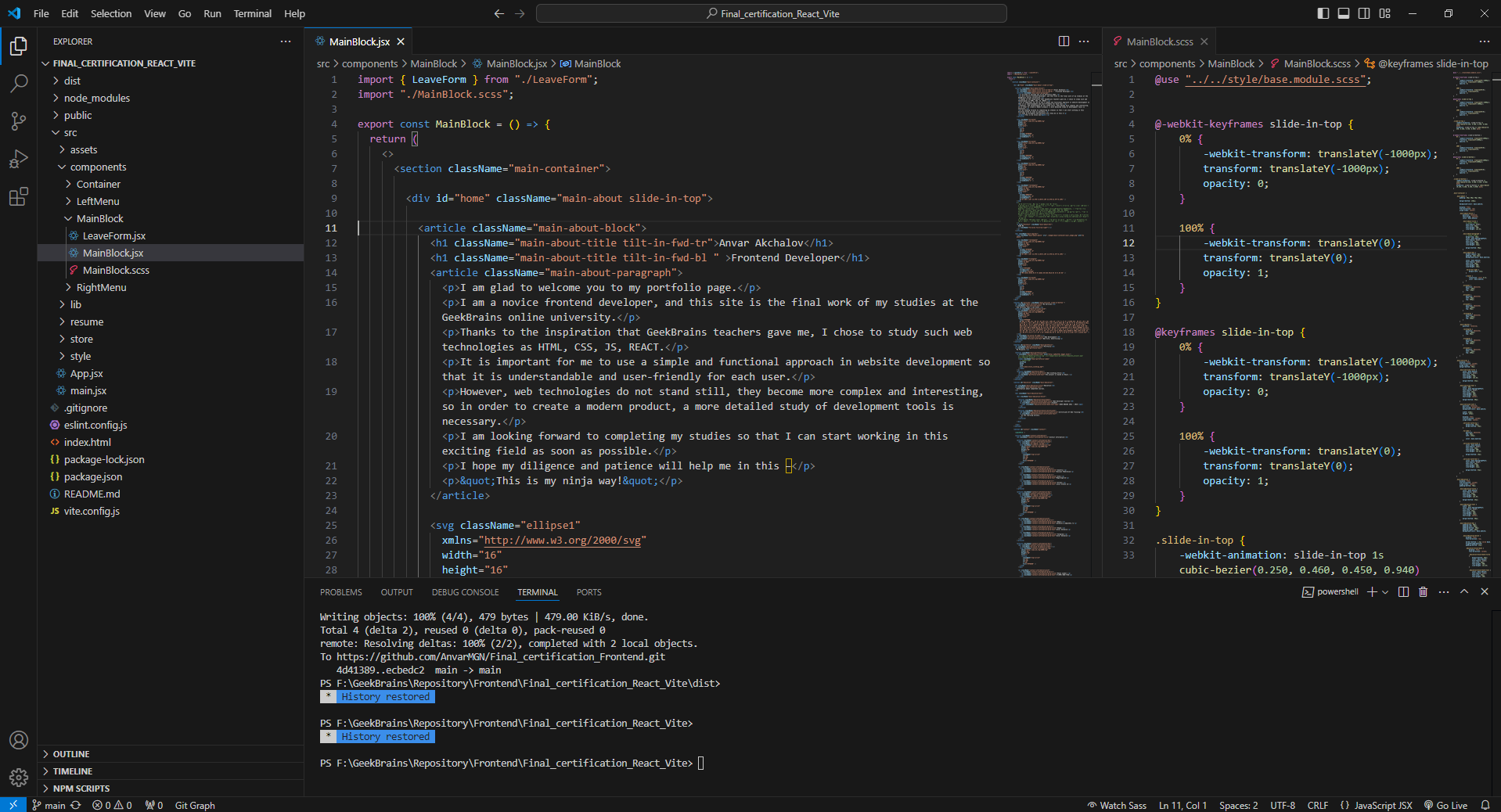


Рисунок 2.1 – Интерфейс Visual Studio Code

Возможности Visual Studio Code:

* Написание и редактирование кода.

Первое и главное назначение редактора — писать исходный код на предпочитаемом языке программирования, редактировать и сохранять его. Так же как редакторы текста оснащены функциями для форматирования контента, редактор кода позволяет структурировать и форматировать программы. Например, он подсвечивает синтаксис, выделяет ошибки, автоматически расставляет отступы. В нем есть и автодополнение: если ввести первые несколько букв команды, он предложит варианты, как можно ее закончить. В VS Code есть целая система для помощи в редактировании кода, ее компоненты сгруппированы под общим названием IntelliSense.

* Автоматическое редактирование.

Редактирование и исправление кода в VS Code можно автоматизировать. Существуют специальные плагины и расширения для редактора, которые автоматически исправляют мелкие ошибки в программе: неправильные отступы, именование, несоответствие стилю кода и так далее. Кстати, стиль кода можно задать в настройках. Конечно, сложную ошибку такой модуль не исправит: он не поймет, как это можно сделать. Но для устранения мелких недочетов возможность удобная.

* Поиск, вставка и другое.

Среди возможностей Visual Studio Code — быстрая навигация по документу и его редактирование. Например, строка поиска поддерживает в том числе регулярные выражения — формулы для разных текстовых сочетаний. С редактированием тоже удобно: можно написать сокращенную формулу той или иной команды, и редактор достроит ее сам. А можно, например, выделить код и закомментировать его одним сочетанием клавиш.

* Контроль версий.

VS Code «из коробки» поддерживает систему контроля версий Git. Прямо из редактора можно «закоммитить» изменения или откатить «коммит», добавить к нему комментарий или отправить в удаленный репозиторий. Для этого не нужно входить в консоль и набирать команды: все делается через визуальный интерфейс. Сделать «коммит» можно в несколько кликов в специальной боковой панели.

* Установка дополнений.

Одна из особенностей VS Code — он очень легко расширяется. Прямо из редактора можно перейти в каталог расширений и дополнений, найти в поиске то, что вам нужно, и установить. Так подключаются поддержка других языков, плагины для автоматического исправления, конфигураторы и много чего еще. Благодаря легкой расширяемости VS Code можно полностью настроить под себя и свои нужды.

* Запуск кода.

Изначально редактор позволяет запускать код только для малого количества языков. Но к нему есть дополнения, которые открывают такую возможность и для других. Чтобы воспользоваться ей, нужно установить соответствующее расширение и перезагрузить VS Code. После этого в интерфейсе редактора должны появиться кнопки для сборки и запуска. Кстати, изначально в нем есть целая вкладка под названием «Запуск и отладка».

* Помощь в отладке.

В VS Code есть встроенный отладчик для языка JavaScript и основанных на нем технологий, таких как TypeScript. Для других языков понадобится устанавливать расширение. Но в любом случае возможность интерактивной отладки в редакторе есть, и это удобно, потому что раньше так «умели» делать только IDE. Интерактивная отладка помогает пошагово выполнять код и на каждом шаге просматривать, как изменяются данные. Так легче обнаружить и устранить ошибку.

Преимущества Visual Studio Code

* Простота освоения.

У редактора интуитивно понятный интерфейс, его легко установить и настроить. А после этого им сразу можно начинать пользоваться. Единственной проблемой поначалу могут стать горячие клавиши и дополнительные настройки — они необычно организованы. Но со временем легко разобраться, особенно если вначале держать под рукой памятку или пользоваться «туториалами».

* Малый вес.

По сравнению с IDE редактор быстро работает и мало весит, не так требователен к ресурсам компьютера. При этом он довольно функционален, а огромное количество расширений позволяет добавлять в него новые возможности.

* Кроссплатформенность.

Visual Studio Code есть для всех популярных операционных систем, так что с ним можно работать вне зависимости от ОС, которую вы предпочитаете. К тому же это позволяет быстрее и удобнее переносить проекты с одного устройства на другое.

* Поддержка разных языков.

Есть инструменты, которые предназначены специально для какого-то языка, например Python или C++. Visual Studio Code поддерживает огромное количество языков программирования, и они легко подключаются. Поэтому с его помощью можно вести проекты, написанные на нескольких языках одновременно.

* Гибкость.

Visual Studio Code имеет множество расширений и удобных функций. Его можно легко и гибко настроить под себя в зависимости от потребностей разработчика и технологий, с которыми он работает.

1.4.3 Git

Git — это система для контроля (управления) версиями исходного кода программ.

В программировании за сохранение кода в контрольных точках отвечает система контроля версий — специальная технология, которую можно подключить к любому проекту. Система контроля версий страхует от ошибок и возвращает код в то состояние, когда всё работало.

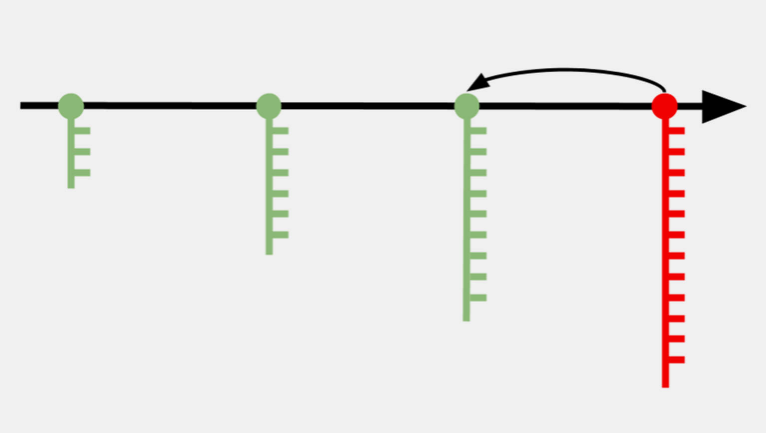
Контрольные точки называются «коммитами». Один «коммит» — это пакет изменений, хранящий информацию с добавленными, отредактированными или удалёнными файлами кода. В один «коммит» принято добавлять не более десяти изменений — так получается длинная история версий, которая позволяет в случае ошибки откатиться с минимальной потерей работоспособного кода (рисунок 3.1). 

Рисунок 3.1 – Система контроля версий — это хранилище с файлами всех изменений проекта. Оно позволяет вернуть проект в любое предыдущее состояние, когда программа стабильно работала.

Как работает Git

Системы контроля версий бывают локальными, централизованными или распределёнными.

Локальная система хранит файлы на одном устройстве, централизованная использует общий сервер, а распределённая — общее облачное хранилище и локальные устройства участников команды. В локальной системе удобно работать с большими проектами, но сложно взаимодействовать с удалённой командой.

В централизованной системе налажена удалённая работа, но всё привязано к одному серверу. Любой сбой или взлом может повредить файлы проекта.

В распределённой системе налажена удалённая работа. Если с файлами основного репозитория что-то случится — проект легко восстановить из копии любого участника команды (рисунок 3.2).

Из-за удобства и гибкости распределённая система версий Git считается современным форматом. Это стандарт для большинства IT-команд.

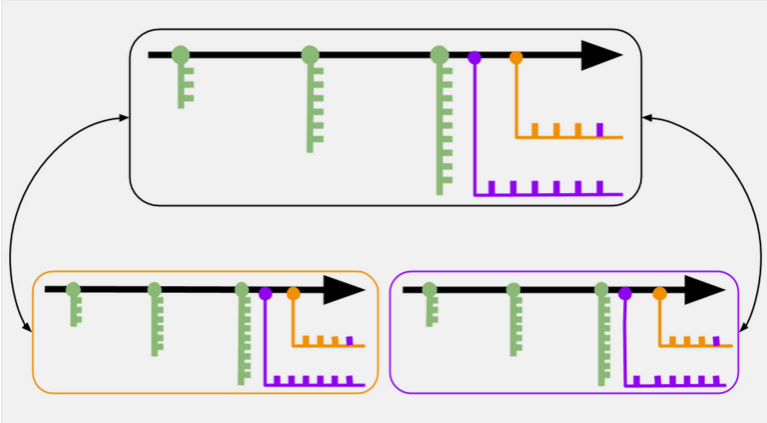


Рисунок 3.2 – Распределённая система контроля версий — это когда у каждого разработчика есть клон проекта

Чем Git отличается от GitHub

Git — это программа, которую нужно установить и подключить к проекту для управления системой контроля версий. GitHub — это сайт-хранилище для историй версий проектов: вы подключаете Git, регистрируетесь на GitHub, создаёте онлайн-репозиторий и переносите файлы из Git на GitHub.

Git — это самая популярная система контроля версий, а GitHub — онлайн-хранилище кода (рисунок 3.3). Git и GitHub настроены на взаимодействие и поэтому часто используются как единый механизм работы с проектом.

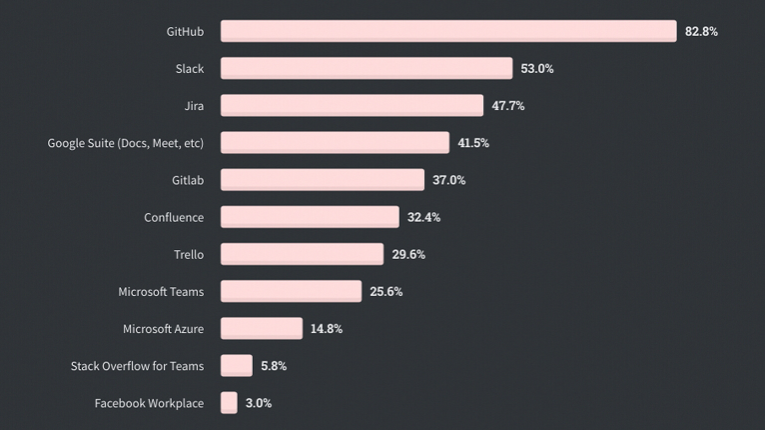


Рисунок 3.3 - GitHub— это самое популярное онлайн-хранилище кода

Если нужно, Git можно заменить альтернативной программой контроля версий, а GitHub — другим онлайн-хранилищем кода. Большинству работодателей это не нужно, поскольку знакомство с другими сервисами отнимает время и неудобно многим разработчикам.

1.4.4 Vite

«Frontend-инструмент нового поколения».

Vite (с французского "быстрый", произносится /вит/) - это инструмент сборки, созданный для обеспечения быстрого и бережливого (lean) процесса разработки современных веб-проектов. Он состоит из двух основных частей:

* Dev сервер, который предоставляет более расширенный функционал, чем нативные ES модули, для примера, невероятно быстрый Hot Module Replacement (HMR).
* Команда сборки, которая связывает ваш код с Rollup, предварительно настроенным для вывода высоко оптимизированных статичных ресурсов (assets) для production.

Vite самодостаточный и имеет практичные настройки по умолчанию прямо из коробки, но он также может очень гибко настраиваться и расширять свой функционал через его Plugin API и JavaScript API с полной поддержкой типов.

Vite предоставляет следующие преимущества:

* Быстрая разработка: Vite использует HMR (Hot Module Replacement), что позволяет разработчикам видеть изменения в коде в режиме реального времени, ускоряя итерации и повышая производительность.
* Небольшие размеры сборки: Vite использует передовые методы сборки, такие как tree-shaking и code-splitting, для создания небольших и оптимизированных пакетов, что улучшает время загрузки и производительность.
* Поддержка современных стандартов: Vite поддерживает последние веб-стандарты, такие как модули ES и CSS-модули, что позволяет разработчикам использовать современные функции и практики.
* Удобство использования с React: Vite специально оптимизирован для использования с React, обеспечивая тесную интеграцию и плавную разработку.

1.4.5 HTML

HTML (от английского HyperText Markup Language) — это язык гипертекстовой разметки текста. Он нужен, чтобы размещать на веб-странице элементы: текст, картинки, таблицы и видео.

Когда вы заходите на сайт, браузер подгружает HTML-файл с информацией о структуре и контенте веб-страницы. HTML как бы выстраивает визуальный фундамент сайта, но не «запускает» сайт самостоятельно. Он всего лишь указывает, где располагаются элементы, какой у них будет базовый дизайн, откуда брать стили для элементов и скрипты (обычно их пишут на JavaScript).

HTML состоит из тегов — команд, которые указывают браузеру, как отображать помещённый в них текст. Это и есть элементы веб-страницы. У каждого тега есть имя, которое заключается в угловые скобки < имя >.

1.4.6 CSS

CSS (Cascading Style Sheets, каскадные таблицы стилей) — язык описания внешнего вида HTML-документа. Это одна из базовых технологий в современном интернете. Практически ни один сайт не обходится без CSS, поэтому HTML и CSS действуют в единой связке.

С помощью CSS можно изменять шрифт, цвет, размер, межстрочный интервал, разделять содержимое на колонки, а также добавлять анимацию и другие декоративные элементы.

Методологии CSS

Нет одной общепринятой методологии CSS. Их несколько (OOCSS, SMACSS, SUIT CSS, Systematic CSS), а со временем возможно появление новых.

Одной из таких методологий является БЭМ. Эта методология была применена в проекте, добавленном в портфолио.

Методология БЭМ была разработана специалистами Яндекса и быстро получила международное признание: в отличие от других CSS-концепций, БЭМ также работает с JavaScript, одним из наиболее популярных решений для создания архитектуры веб-приложений.

В основе БЭМ — соглашение по именованию, которое делает имена CSS-классов максимально информативными для разработчиков и позволяет многократно использовать компоненты. Подход упрощает командную работу, масштабирование и поддержку проектов.

БЭМ — компонентный подход к веб-разработке, в основе которого лежит принцип разделения интерфейса на независимые блоки, а также набор интерфейсных библиотек, фреймворков и вспомогательных инструментов. Расшифровывается как «Блок, Элемент, Модификатор».

Зачем нужен БЭМ

Соблюдение правил методологии ускоряет процесс разработки:

* Упрощает код и облегчает рефакторинг;
* Избавляет от необходимости комментировать код, поскольку назначение CSS-селекторов становится очевидным по названиям;
* Позволяет использовать одни и те же фрагменты кода многократно, в том числе на одном DOM-узле;
* Предотвращает взаимное влияние компонентов друг на друга.

Рассмотрим блоки, элементы и модификаторы подробнее.

Блок

Блок — логически и функционально независимый компонент страницы: меню, боковая панель, карусель, шапка сайта и так далее. Блок включает в себя шаблоны (Pug, Handlebars) и CSS-стили, скрипты JavaScript, документацию в формате XML или Markdown, другие необходимые для реализации технологии.

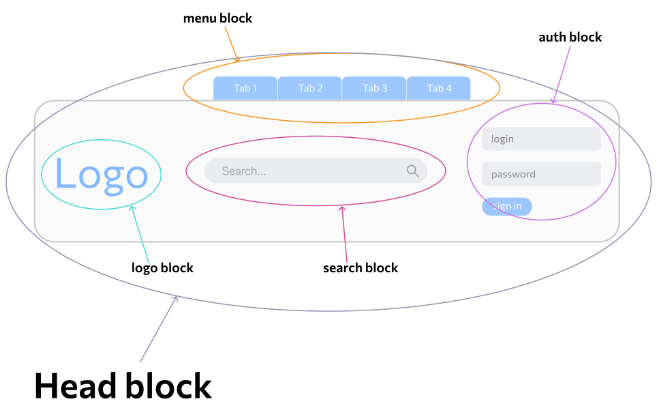


Рисунок 6.1 - Блок

Шапка сайта инкапсулирует навигационное меню, логотип, поиск и авторизацию (рисунок 6.1).

Независимость блоков позволяет свободно перемещать их в пределах страницы и всего проекта. Составные части блока, например формы авторизации и поиска, можно поменять местами: они будут корректно работать и сохранят внешний вид. Внесение изменений в CSS- или JavaScript-скрипты форм не потребуется.

Благодаря верстке по БЭМ на страницах можно размещать сколько угодно экземпляров одного и того же блока.

Элемент

Элемент — неотделимая составная часть блока. Особенности элементов:

* Они не существуют и не используются вне блока;
* Принадлежат только одному блоку;
* Могут вкладываться друг в друга.

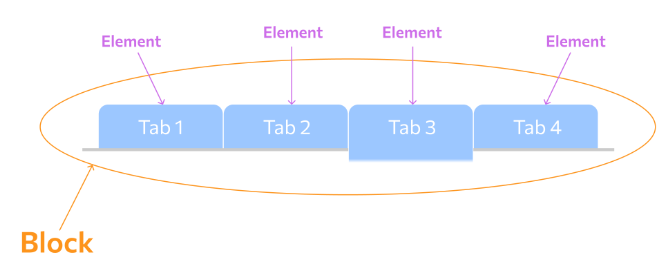


Рисунок 6.2 - Элемент

Вкладка принадлежит блоку меню и не может использоваться вне блока (рисунок 6.2).

Это необязательные компоненты: небольшие блоки могут обходиться без вложенных элементов. Методология не устанавливает жестких правил в отношении того, когда разработчику следует использовать блок, а когда — элемент. Блоки, как правило, нужно создавать в том случае, если фрагмент кода может использоваться самостоятельно. Зависимым фрагментам кода (элементам) необходима родительская сущность — блок.

Рекомендации БЭМ-верстки касаются смысла и структуры наименования элементов:

Название элемента должно характеризовать его смысл, роль — item («пункт меню»), link («ссылка»).

Структура имени элемента должна соответствовать схеме имя-блока\_\_имя-элемента — имя элемента отделяется двойным подчеркиванием.

Модификатор

Модификатор — это сущность, которая определяет внешний вид, состояние и поведение элемента или блока. Один и тот же блок, например, меню будет выглядеть по-разному в зависимости от применяемых модификаторов (рисунок 6.3).

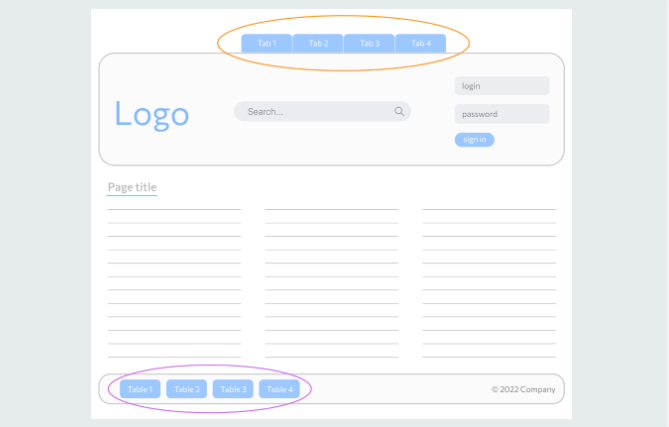


Рисунок 6.3 - Внешний вид верхнего и нижнего меню заметно отличается.

Модификаторы делятся на два типа:

1. Булевые. Применяют, когда факт наличия модификатора важнее, чем его значение. Например, если элемент имеет модификатор disabled («отключен»), значение по умолчанию равно true. Структура имени при использовании булевого модификатора выглядит так: имя-блока\_\_имя-элемента\_имя-модификатора.
2. Ключ-значение. Используют в тех случаях, когда значение модификатора важно. Например, если применяется блок с определенной темой оформления: search-form\_theme\_forest. Структура наименования модификатора элемента соответствует схеме: имя-блока\_\_имя-элемента\_имя-модификатора\_значение-модификатора.

Модификаторы могут изменяться в процессе работы блока в зависимости от запросов из других блоков или как реакция на DOM-события. Например, если пользователь введет неверные данные в форму авторизации и нажмет кнопку «Войти», DOM-событие click («Нажатие») изменит свойство блока с сообщением об ошибке с невидимого на видимое.

Наличие модификаторов у элементов и блоков — опционально, количество — не ограничено. Следует помнить, что блоку и его элементу нельзя присваивать разные значения одного и того же модификатора. По правилам БЭМ, название модификатора должно характеризовать:

* Внешний вид блока или элемента;
* Состояние компонента;
* Поведение.

Традиционно имя модификатора отделяют от имени блока или элемента одним подчеркиванием (\_). В качестве альтернативы также используют двойной дефис (—).

Данная методология частично применена в этом проекте и полностью использована в проекте, находящемся в разделе «Portfolio».

1.4.7 SASS

Sass — это метаязык (язык для описания другого языка), который упрощает и ускоряет написание CSS-кода. Его часто называют препроцессором CSS — это означает, что Sass имеет свой синтаксис, из которого генерирует CSS-код, понятный любому браузеру.

Для чего используют Sass

Препроцессор Sass помогает:

* Сделать CSS-код понятнее и проще. Его легче масштабировать, обновлять и поддерживать;
* Расширить функциональность. С помощью Sass можно использовать CSS-константы, встроенные функции, вложенные правила, примеси (смешанные стили), наследование, математические операторы;
* Избежать многократного повторения одинаковых фрагментов кода. Это экономит время разработчика, уменьшает объем файлов стилей и ускоряет обработку страниц.

Браузеры не понимают код Sass — они распознают и отображают только стили CSS. Для преобразования используется специальное приложение — препроцессор, который переводит конструкции в понятный браузеру формат. В готовый CSS-код не включаются построчные комментарии. В ходе компиляции в CSS передаются только стандартные примечания, отделяемые от кода символами /\* \*/.

Примеры использования препроцессора можно отследить в файлах стилей проекта.

1.4.8 JavaScript

JavaScript (англ. /ˈdʒɑːvəskrɪpt/; аббр. JS) — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java. Языком JavaScript не владеет какая-либо организация или компания, что отличает его от ряда языков программирования, используемых в веб-разработке.

Название «JavaScript» является зарегистрированным товарным знаком корпорации Oracle в США.

1.4.9 React

React.js — это библиотека для языка программирования JavaScript с открытым исходным кодом для разработки пользовательских интерфейсов. Она помогает быстро и легко реализовать реактивность — явление, когда в ответ на изменение одного элемента меняется все остальное.

React имеет открытый исходный код и мощное сообщество. Это одна из самых популярных библиотек для веб-разработки.

Для чего нужен React

React используют для создания одностраничных и многостраничных приложений, разработки крупных сайтов. Например, с его помощью написан стриминговый сервис Netflix и реализованы новостные ленты крупнейших социальных сетей. Библиотека предназначена:

* Для создания функциональных интерактивных веб-интерфейсов, работая с которыми, не нужно постоянно обновлять страницу;
* Быстрой и удобной реализации отдельных компонентов и страниц целиком — элементы в React легко использовать повторно;
* Легкой разработки сложных программных структур — их просто описывать, если использовать реализованный в React подход;
* Доработки новой функциональности с любым изначальным стеком технологий: библиотека не зависит от остального инструментария и будет хорошо работать, на чем бы ни был написан код;
* Разработки одностраничных и многостраничных приложений (SPA и PWA). Это приложения, которые функционируют как программы и веб-сервисы и имеют соответствующий интерфейс;
* Работы с серверной частью сайта или разработки мобильных приложений. В таких случаях React используют совместно с инструментами, адаптирующими веб-технологии под другие цели.

Особенности библиотеки React

React не является полноценным фреймворком, являясь библиотекой для создания пользовательских интерфейсов. Важно понимать разницу между фреймворками и библиотеками:

1. Фреймворк — это комплексное решение, которое предоставляет набор инструментов, библиотек и правил для разработки приложений. Фреймворк часто определяет структуру приложения, а разработчику предоставляется меньше свободы в выборе инструментов и архитектурных решений. Примеры фреймворков включают Angular и Ember.
2. Библиотека — это набор функций и компонентов, которые помогают в определенных задачах, но не навязывают общую архитектуру приложения. Разработчик имеет большую свободу выбора остальных инструментов и архитектурных решений. React — это пример библиотеки, которая сосредотачивается на создании пользовательских интерфейсов.

Таким образом, React предоставляет инструменты для создания компонентов пользовательского интерфейса, управления состоянием и взаимодействия с DOM, но не навязывает конкретную архитектуру приложения. Разработчики могут использовать React в комбинации с другими библиотеками и инструментами по своему усмотрению, что делает его гибким и позволяет создавать разнообразные типы веб-приложений.

Декларативность

Декларативный стиль означает, что достаточно один раз описать, как будут выглядеть результаты работы кода — элементы в разных состояниях. Ему не нужно фокусироваться на способах достижения результатов: большую часть задач выполнит библиотека. React.js будет автоматически обновлять элементы в зависимости от условий, главная задача — грамотно описать их. Удобный и понятный подход облегчает написание и отладку кода.

Виртуальное DOM-дерево

Любой веб-интерфейс основан на HTML-документе и CSS-стилях, к которым подключен код на JavaScript. Структура HTML-документа, точнее его модель, называется DOM-деревом (DOM расшифровывается как Document Object Mode, объектная модель документа). Это древовидная модель, в которой в иерархическом виде собраны все используемые на странице элементы.

React включает в себя механизм, который управляет процессом рендеринга компонентов. Рендеринг — это процесс создания визуального представления компонента, которое отображается в браузере или мобильном приложении.

Особенность React в том, что он создает и хранит в кэше виртуальное DOM-дерево — копию DOM, которая изменяется быстрее, чем реальная структура. Это нужно для того, чтобы быстро обновлять страницы. Если пользователь выполнит действие или наступит какое-либо событие, DOM должна измениться, так как изменятся объекты на странице. Но реальная объектная модель может быть огромной, ее обновление — медленный процесс. Поэтому React работает не с ней, а с виртуальной копией в кэше, которая весит меньше.

Когда происходит событие, из-за которого код должен обновить объект, изменение быстро отображается в виртуальном DOM. После этого обновляется реальная объектная модель. Для пользователя это значит, что изменения на странице отобразятся мгновенно, а не после долгой загрузки.

Обновление DOM по частям

Чтобы улучшить быстродействие, React обновляет DOM не полностью. Он хранит в памяти две облегченных копии: актуальную и предыдущую. Когда что-то обновляется, библиотека сравнивает версии между собой и изменяет только ту часть дерева, которая действительно поменялась. Это нужно, чтобы не перезагружать DOM целиком и не замедлять страницу. Подход кажется сложным, но он важен для оптимизации загрузки.

Возможность повторно использовать компоненты

React основан на компонентах — отдельных элементах веб-интерфейса. Компоненты инкапсулированы, то есть самостоятельны: в каждом из них размещены все необходимые методы и данные. Подробнее про инкапсуляцию можно почитать в нашей статье про ООП: это один из ключевых критериев объектно-ориентированного подхода. В случае с компонентами React инкапсуляция также означает, что состояние элемента хранится в нем самом, а не в отдельном объекте.

Инкапсулированные самостоятельные компоненты можно использовать повторно, размещать в другом месте кода, в ином разделе или на другой странице. Данные можно переносить по всему приложению, использовать вне DOM конкретной страницы. Это ускоряет разработку и сокращает количество действий для создания функционирующего интерфейса. Благодаря отсутствию сложных зависимостей инкапсуляция также облегчает отладку.

Нисходящий поток данных

Компоненты могут передавать свойства и данные друг другу, но только в одном направлении — от «родительских» к дочерним. Это помогает реализовать четкую иерархию, облегчает отладку. Однонаправленный поток данных означает, что программист всегда может понять, откуда именно к элементу поступили данные.

Еще одна особенность потока данных — отсутствие возможности изменять свойства напрямую. Для этого существуют специальные callback-функции. Так называются части исполняемого кода, которые передаются от одного компонента другому в виде параметра функции. Это позволяет поддерживать стабильность: свойства оказываются неизменяемыми после создания, а callback-функция как бы пересоздает их. В результате использования коллбэков поток данных выглядит так: свойства и информация передаются сверху вниз, а события происходят снизу-вверх.

Синтаксис JSX

JSX расшифровывается как JavaScript XML. Это расширение языка JavaScript, которое помогает описывать HTML-подобные элементы с помощью кода на React. С помощью синтаксиса на React создают компоненты страницы и гибко управляют ими.

Несмотря на то что элементы похожи на HTML, это по-прежнему язык JavaScript с возможностью быстро и легко изменять DOM с помощью кода. И все же JSX воспроизводится как HTML: по сути разработчик описывает нужный компонент на языке разметки, а тот остается JavaScript-объектом с широкой функциональностью. Это удобно, упрощает программирование, но может запутать начинающих.

React Hooks

В старых версиях управлять состояниями можно было с помощью классов — специальных конструкций, о которых можно подробнее прочесть в статье про ООП. Сейчас в React.js есть поддержка хуков — так называются специальные функции-«крючки», которые «цепляются» за состояние элемента или за метод. Изменение состояния или вызов метода «тащит» за собой эти функции, и они автоматически выполняются — это помогает избежать использования классов, облегчает и упрощает написание кода.

В React есть встроенные хуки, но разработчик может создавать и собственные — это помогает повторно использовать код и уменьшить его количество.

React Developer Tools

Так как React.js — очень популярная технология, его создатели разработали бесплатные расширения для браузера с инструментами для проверки и отладки. Фронтендеры часто пользуются консолью и панелью разработчика в браузере, чтобы проверить, как работает их код. React Developer Tools облегчают задачу и расширяют возможности. Например, можно просматривать прямо в браузере компоненты с большим уровнем вложенности и не искать их в коде долгое время. React Developer Tools существуют для новых версий популярных браузеров Firefox и Google Chrome.

Преимущества React

* Популярность

Это один из трех самых распространенных инструментов для фронтенд-разработки. Кроме него, популярны Vue.js и Angular, но первый пока не так распространен, а второй намного сложнее в изучении. Еще одна популярная технология — jQuery, но она постепенно уходит в прошлое. На React написано множество сайтов и приложений. Ее используют такие крупные компании, как Яндекс, Uber, Сбербанк, Авито, BBC, Airbnb, Netflix и другие. Вакансий много, и они высоко оплачиваются даже на начальных уровнях.

* Огромное сообщество

Это еще одно следствие популярности: React-разработчиков много, в том числе опытных, многие из них пишут туториалы и статьи, помогают новичкам влиться в процесс разработки. В ходе обучения начинающий всегда может обратиться на тематические ресурсы, где ему постараются помочь. Людей в сообществе хватает, можно получить ответы на большинство вопросов. Документация поддерживается и обновляется, существует в том числе на русском языке.

* Развитая экосистема

Создатели React активно побуждали пользователей вкладываться в развитие проекта, и энтузиасты писали свои технологии для совместного использования с библиотекой. В результате сейчас сформировалась целая экосистема: десятки библиотек и других инструментов, которые можно использовать в разработке. Это делает процесс проще и легче, позволяет решать огромное количество задач.

* Простота создания

С помощью компонентов проекта можно быстро и легко собрать интерактивный, чутко реагирующий на любые изменения интерфейс сайта или приложения любой сложности. Это намного проще, чем прописывать реакции на всевозможные события вручную, и снижает количество ошибок. Так как компоненты можно использовать повторно в любом месте кода, задача становится еще легче. Например, для создания сложного многостраничного сайта или приложения не нужно многократно писать один и тот же компонент.

* Реактивность

Это одно из ключевых преимуществ проекта, вынесенное в название. Библиотека реагирует на обновление компонента и автоматически отображает его изменения в дереве документа. Изменения могут происходить в ответ на действия пользователя, какие-то внешние перемены или другие события. В результате сайты и приложения становятся более привлекательными для пользователя.

* Эффективность

Благодаря виртуальному DOM библиотека экономит ресурсы. Чтобы изменить состояние элементов, не нужно полностью перезагружать все DOM-дерево, лишний раз тратить трафик пользователя и нагружать браузер. Изменяются только конкретные элементы, это происходит через виртуальное DOM-дерево — сайты и приложения становятся «легче» и удобнее.

* Высокая скорость

Сайты и приложения, написанные на React, быстрые и отзывчивые благодаря виртуальному DOM. Дерево занимает меньше места, быстрее обновляется.

Также ускоряется работа программистов. Компоненты легко создавать и переиспользовать, между ними уже продумана логика сообщения. Главная задача — грамотно описать состояния. Не нужно прописывать всю логику и принципы с нуля, это за разработчика уже сделали создатели React.

* Удобная отладка

Ошибок с React бывает меньше, чем при написании кода на «чистых» языках без библиотек. Это происходит, потому что многое реализуется автоматически, соответственно, уменьшается влияние «человеческого фактора». Но даже если при разработке программист допустил ошибку, ее довольно легко отследить и исправить благодаря четкой логике потока данных, дополнительным инструментам, понятному синтаксису. Код на React легко читать, понимать и отлаживать.

1.4.10 Redux

Библиотека управления состоянием Redux

Управление состоянием является одной из самых больших проблем при использовании фреймворка React. Это касается не только пользователей. Разработчикам нужен простой и масштабируемый процесс управления состоянием для проектирования эффективных и сложных пользовательских интерфейсов.

Обычно используют «хуки» React для доступа и обмена состояниями между разными компонентами. Но при работе с их значительным количеством сложность становится слишком большой. В таких случаях необходимо использовать библиотеки управления состоянием.

Что такое управление состоянием в React

Состояние представляет собой часть компонента, которая изменяется в зависимости от действий пользователя. Это объект JavaScript, который служит памятью компонента и хранит многие его свойства. Таким образом, управление состоянием в React — это процесс обмена данными между различными компонентами.

Когда пользователь взаимодействует с Вашим React приложением, в состоянии одного или нескольких компонентов могут происходить изменения. Эти изменения могут повлиять на интерфейс (UI), представленный пользователю, поэтому Вам необходимо управлять ими.

Почему управление состоянием важно в React

Управление состоянием чрезвычайно важно для сохранения целостности приложения, так как оно обеспечивает простоту взаимодействия с пользователем.

Когда пользователи взаимодействуют с вашим React приложением, каждое их действие может изменить состояние нескольких компонентов. Возьмем, к примеру, приложение интернет-магазина, в котором покупка товара может повлиять на несколько компонентов посредством следующих действий:

* Добавление товаров в корзину;
* Добавление товаров в историю заказов покупателя;
* Изменение количества купленных товаров.

В таком сценарии со сложными элементами, взаимодействующими друг с другом и требующими обмена состояниями между собой, легко нарушается масштабируемость. Если не внедрены надёжные библиотеки управления состоянием, задача частой отладки и исправления приложения может быстро стать нереализуемой.

Управление состоянием имеет важное значение для обеспечения простой, согласованной и масштабируемой передачи состояний между несколькими компонентами.

Что из себя представляют библиотеки управления состоянием в React

Библиотеки управления состоянием предлагают новые подходы, делая это эффективно и помогая разработчикам создавать масштабируемые и интерактивные приложения на React.

Эти библиотеки представляют собой модульные фрагменты кода, которые учитывают передовые методы управления состоянием. Освоить их легко и просто. И у разработчиков есть широкий выбор как из проверенных, так и из новых.

Различные библиотеки используют разные подходы для достижения одного и того же результата. Они также отличаются друг от друга размером, поддержкой языков, документацией, поддержкой API и многим другим.

При выборе правильной библиотеки для вашего приложения на React вы должны учитывать следующие факторы:

* Удобство использования;
* Производительность;
* Масштабируемость;
* Модифицируемость;
* Сложность поддержки;
* Возможность переиспользования;
* Тестируемость;
* Экосистема;
* Сообщество.

На GitHub есть более 90 библиотек управления состоянием для React, ниже приведены топ-7 библиотек (рисунок 10.1) отвечающих перечню требований из списка и пользующихся большой популярностью среди разработчиков.

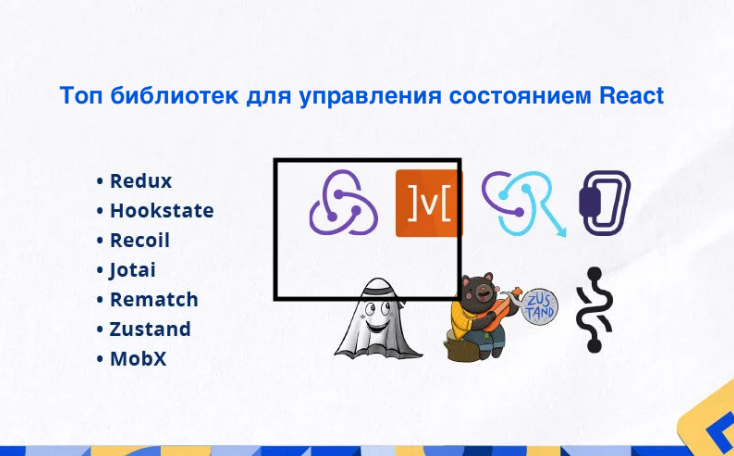


Рисунок 10.1 – Топ -7 библиотек управления состоянием приложения

Redux

Redux существует с 2015 года и является первой библиотекой управления состоянием в React, созданной для упрощения этого процесса путем централизации хранилища состояний, которая достигается с помощью использования «Store», содержащего все состояния вашего приложения.

По результатам npmtrends на январь 2022 года, Redux по-прежнему занимает лидирующую позицию среди остальных подобных библиотек. В Redux используются «экшены» (actions) и «редьюсеры» (reducers). «Экшены» — это объекты, сообщающие хранилищу (store) о том, какие события должны произойти. «Редьюсеры» — функции, которые выполняются на основе входных данных экшена и начального состояния (initial state).

Всякий раз, когда необходимо изменить состояние компонента, зависящее от поведения пользователя, вы можете отправлять (dispatch) «экшен» «редьюсеру», который изменяет состояние и сохраняет его в хранилище.

Хотя Redux широко популярен, разработчики не очень любят работать с шаблонным кодом, который тесно с ним связан. Множество «экшенов» и «редьюсеров» приводят к большому количеству кода, который становится трудно поддерживать, когда приложения становятся сложными. Для уменьшения шаблонных строк и общего упрощения можно использовать Redux Toolkit.

Благодаря тому, что Redux Toolkit заботится об удобстве использования Redux, простая логика и чистые функции делают его довольно удобным в сопровождении, производительным и тестируемым. Вы можете изменять и повторно использовать код Redux, так как он не зависит от фреймворка и поддерживает промежуточный слой (middleware).

Redux, благодаря своей зрелости имеет поддержку большого сообщества разработчиков и богатую экосистему дополнений и инструментов.

Для управления состоянием формы ввода в проекте, используется именно Redux.

1.4.11 Веб-браузеры и браузерные инструменты разработчика

Для работы с проектом использовались браузеры Opera и Chrome.

Оба браузера, предлагают пользователям быстрый и безопасный просмотр веб-страниц. Однако есть несколько ключевых различий:

* Встроенные функции: Opera поставляется с рядом встроенных функций, таких как блокировщик рекламы, VPN и рабочие пространства, которые могут быть полезны для некоторых пользователей.
* Экосистема расширений: У Chrome более обширная экосистема расширений, что дает пользователям больший выбор для настройки и расширения функциональности браузера.
* Ресурсоемкость: Opera потребляет меньше ресурсов, чем Chrome, что делает ее лучшим выбором для компьютеров с ограниченными ресурсами.

В конечном итоге выбор между Opera и Chrome зависит от индивидуальных предпочтений и потребностей пользователя. Оба браузера являются отличными вариантами для просмотра веб-страниц, и каждый из них предлагает свой уникальный набор функций.

Для тестирования и отладки сайта-портфолио использовались следующие браузерные инструменты разработчика:

* Интерактивный поиск: Быстро ищите элементы на странице, используя селекторы CSS или XPath.
* Инспектор элементов (Elements): Позволяет инспектировать DOM, стили и события страницы.
* Редактор стилей (Styles): Редактируйте стили в режиме реального времени и сразу же просматривайте изменения.
* Консоль (Console): Позволяет выполнять код JavaScript, оценивать выражения и просматривать сообщения об ошибках.
* Отладка по шагам (Sources): Позволяет просматривать и редактировать исходный код страницы, а также устанавливать точки остановки выполнения кода и отлаживать сценарии.
* Lighthouse: Позволяет запускать аудиты страниц для проверки производительности, доступности и других аспектов.
* Эмуляция устройств (Toggle device toolbar): Просматривайте страницу на разных устройствах и разрешениях экрана.
  1. Вывод по главе

В результате проведённого анализа были рассмотрены современные методы frontend-разработки. Описаны инструменты и технологии frontend-разработки, которые будут применены в проекте и приведены аргументы в пользу их использования.

Так же выбран макет Figma (приложение А) для разработки сайта-портфолио с учётом выявленных ключевых характеристик.

1. РАЗРАБОТКА ФРОНТЕНД ЧАСТИ САЙТА
   1. Создание репозитория

Для сохранения прогресса был создан удалённый онлайн-репозиторий на GitHub - <https://github.com/AnvarMGN/React-JS>.

Далее создан локальный репозиторий с добавлением в нём папки для проекта Final\_certification и настроена работа между удалённым и локальным репозиториями:

* git remote add origin https://github.com/AnvarMGN/ React-JS.git;
* git branch -M main;
* git push -u origin main.
  1. Развёртывание проекта

Развёртывание проекта React осуществляется с использованием инструмента сборки Vite:

* npm create vite@latest final\_project\_on\_vite

Select a framework: » React,

Select a variant: » JavaScript

* npm install – установка зависимостей
* "scripts": { "dev": "vite --open", } – настройка автоматического запуска локального сервера в браузере
* npm install -D sass-embedded --- vite – установка препроцессора SASS
* npm install redux @reduxjs/toolkit – установка библиотеки управления состоянием Redux
  1. Архитектура проекта.

Проект имеет следующую структуру:

* node\_modules;

Директория node\_modules предназначена для хранения библиотек, использованных в разработке. Библиотеки и фреймворки, которые указаны в файле package.json, устанавливались с помощью пакетного менеджера NPM.

* public;

В директории public размещены изображения и видео файлы проекта, хранящихся соответственно в папке images и video.

* src;

Директория src, предназначена для хранения основной части проекта и имеет следующую структуру:

* components – компоненты интерфейса;
* lib – хранит библиотеку шрифтов FontAwesome;
* resume – хранит тест-пример файл резюме my\_cv. pdf;
* store – стейт-менеджмент приложения;
* style – хранит файл стилей base.module.scss;
* App.jsx – базовый компонент, который служит контейнером для всех других компонентов, используемых в приложении;
* Main.jsx – точка входа в приложение с идентификатором root из файла index.html.
  1. Создание компонентов JSX

2.4.1 Компонент Container.jsx

Общий компонент-контейнер Container представлен на рисунке 4.1.



Рисунок 4.1 – Компонент Container

Как видно на рисунке 4.1 в него импортируются компоненты LeftMenu, MainBlock, RightMenu, а также файл стилей Container.scss.

Файл стилей Container.scss хранит настройки, отменяющие форматирование по умолчанию – списков, ссылок, отступов; центрирует контейнер с помощью класса ".center", а так же добавляет плавность при прокрутке блоку ". main-container" когда пользователь использует меню навигации.

В файл стилей Container.scss импортируются шрифты FontAwesome и файл base.module.scss, который в качестве переменных хранит цвета используемые в проекте (рисунок 4.2).

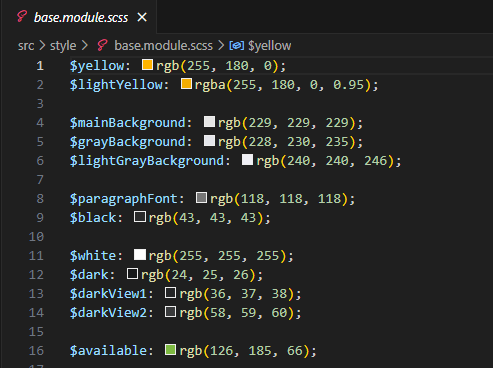


Рисунок 4.2 – Файл стилей base.module.scss

2.4.2 Компонент LeftMenu.jsx

Компонент LeftMenu.jsx (рисунок 4.3) включает в себя вёрстку левого фиксированного блока Left menu из макета Figma (приложение А) и компонент кнопки «DOWNLOAD CV» для скачивания резюме DownloadResumeButton.jsx.

В компонент LeftMenu.jsx импортируется файл стилей LeftMenu.scss. LeftMenu.scss описывает стили для применённых шрифтов, размеров элементов, позиционирования, эффектов наведения и анимации.

Для компонентов MainBlock, RightMenu будет использована аналогичная схема применения файла стилей.



Рисунок 4.3 – Компонент LeftMenu.jsx

Компонент DownloadResumeButton.jsx (рисунок 4.4) поддерживает следующий функционал:

1. Компонент использует «хук» useState для хранения URL-адреса файла резюме (cvUrl) в состоянии компонента.
2. Функция handleDownload вызывается при нажатии на кнопку. Она выполняет следующие действия:

* Создает элемент a (link) в DOM;
* Устанавливает атрибут href для ссылки link на cvUrl, чтобы указать местоположение файла резюме;
* Устанавливает атрибут "download" и "resume.pdf";
* Добавляет ссылку link в document.body;
* Имитирует клик по ссылке, что запускает загрузку файла;
* Удаляет ссылку из document.body, чтобы очистить DOM после загрузки.

1. Компонент возвращает JSX, который отрисовывает пользовательский интерфейс кнопки.

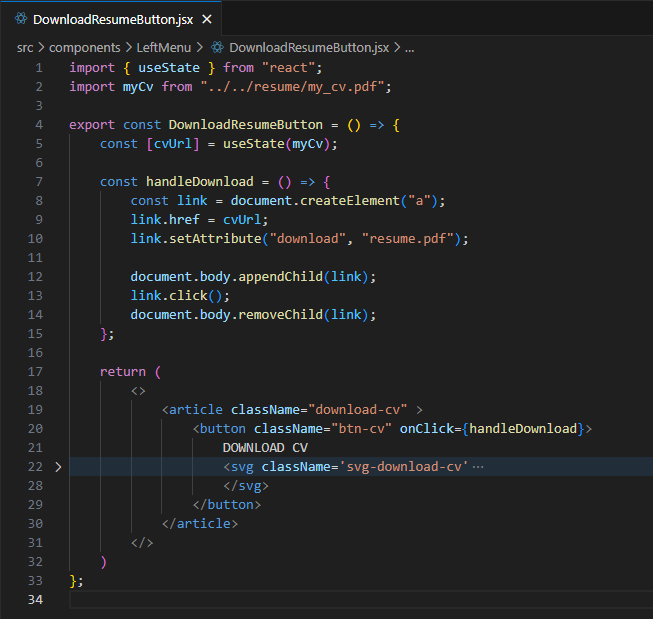


Рисунок 4.4 – Компонент DownloadResumeButton.jsx

2.4.3 Компонент MainBlock.jsx

Компонент MainBlock.jsx имеет следующую структуру (рисунок 4.5).

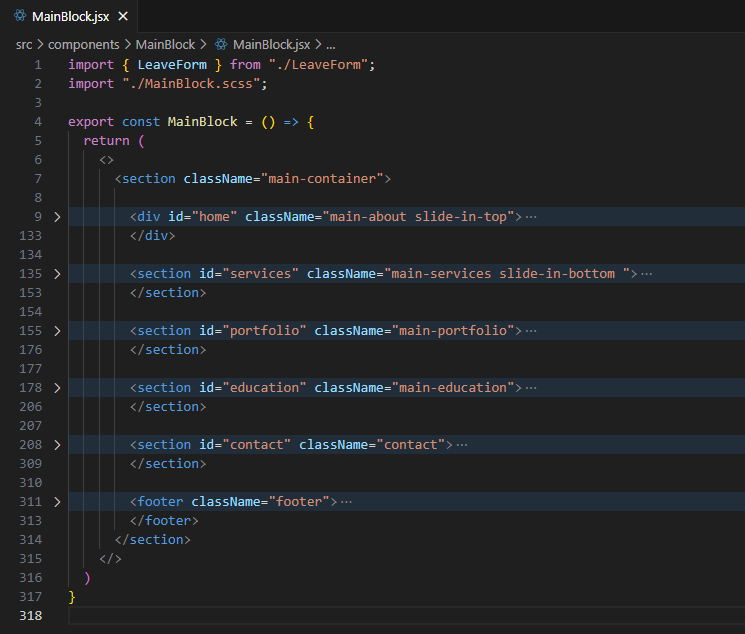


Рисунок 4.5 - Компонент MainBlock.jsx

Каждому разделу присвоен идентификатор, который позволяет использовать меню навигации из компонента RightMenu.

В разделе с идентификатором portfolio (рисунок 4.6) и классом ".main-portfolio" в карточке проекта добавлено видео с тегом <video> и классом ".main-portofolio-video", которое автоматически воспроизводится, зацикливается и отключает звук. Видео имеет атрибут src, указывающий на файл video/store\_clothing.mp4.

При нажатии на ссылку видео приостанавливается, а пользователь перенаправляется на целевую страницу <http://p91277dr.beget.tech/>, где расположен еще один проект, подробнее о нём будет описано в разделе 2.5.



Рисунок 4.6 – Карточка проекта портфолио

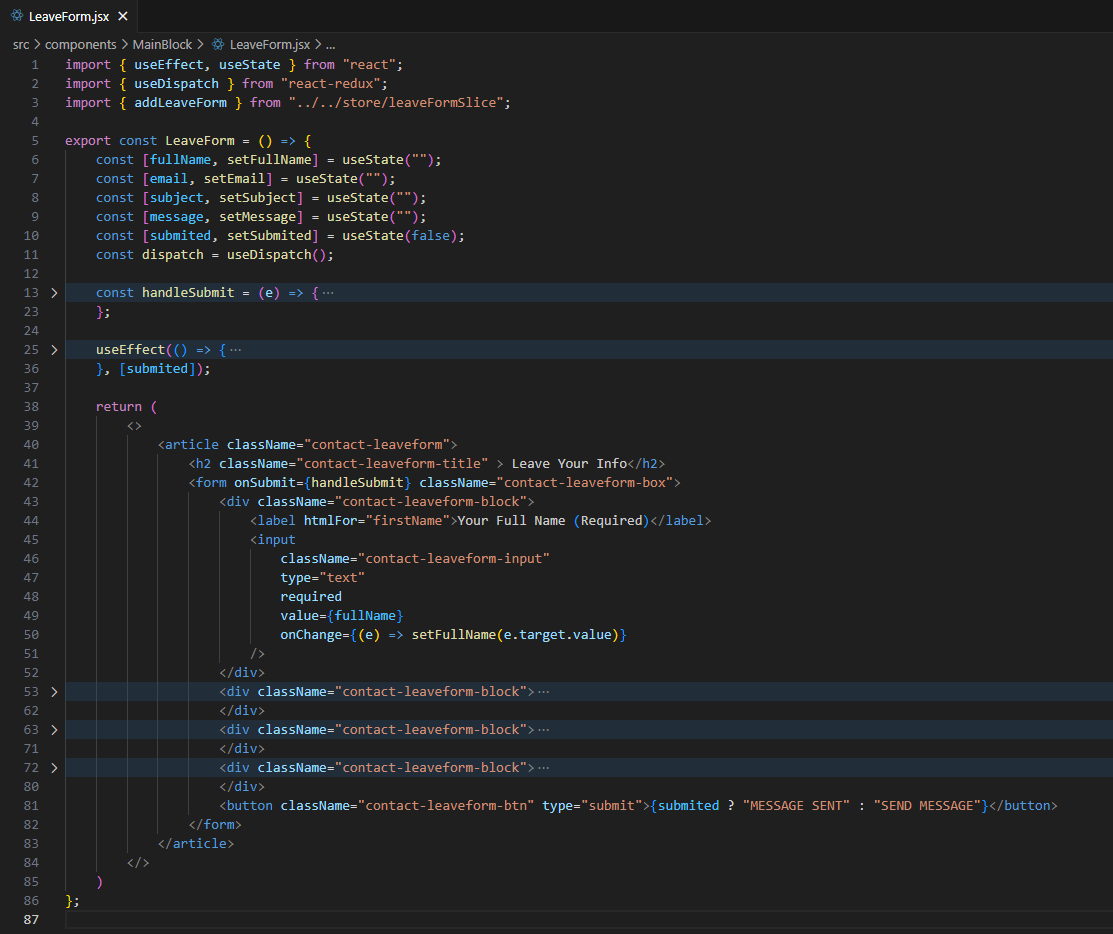
В раздел с идентификатором "contact" импортирован компонент LeaveForm.jsx (рисунок 4.7).

Компонент использует Redux для управления состоянием формы. Функция addLeaveForm отправляет данные формы в Redux store.

Компонент LeaveForm.jsx поддерживает следующий функционал:

1. Компонент использует «хуки» useState для хранения состояния формы, включая полное имя, электронную почту, тему и сообщение. Он также использует состояние "submited" для отслеживания отправки формы.
2. Функция handleSubmit вызывается при отправке формы. Она отправляет данные формы в Redux store с помощью функции addLeaveForm. После отправки формы она очищает состояние формы и устанавливает "submited" в true.
3. «Хук» useEffect используется для отображения сообщения «MESSAGE SENT» в течение 4 секунд после отправки формы.
4. Компонент возвращает JSX, отрисовывающий форму:

* Поля ввода для имени, электронной почты, темы и сообщения;
* Кнопка отправки с текстом «SEND MESSAGE», который меняется на «MESSAGE SENT» после отправки формы.

Рисунок 4.7 – Компонент LeaveForm.jsx

* + 1. Компонент RightMenu.jsx.

Компонент RightMenu.jsx имеет простую структуру состоящую из пяти навигационных ссылок с иконками, при наведении на которые отображается название раздела к которому относится ссылка. При нажатии на иконку, блок c классом ".main-container "прокручивается до выбранного раздела.

* 1. Краткое описание проекта магазина одежды – Web Clothing Store

Web Clothing Store – представляет собой интернет-магазин одежды, разработанный с использованием фреймворка React.

Интернет магазин имеет следующий функционал:

* Товары с разных страниц магазина можно добавлять в корзину;
* Количество товаров в корзине можно изменять, а также удалять, при нажатии на соответствующую кнопку;
* Реализован подсчёт общей стоимости корзины;
* Добавлена фильтрация товаров по выбранному размеру.

Сборка проекта размещена на веб-хостинге Beget по адресу: <http://p91277dr.beget.tech/> .

* 1. Сборка и размещение сайта-портфолио на GitHub Pages

Перед тем как разместить проект на веб-хостинге была произведена предварительная подготовка, а затем сборка приложения с помощью инструментов Vite.

На этапе подготовки был создан новый репозиторий на GitHub (<https://github.com/AnvarMGN/Final_certification_Frontend>) и отредактирован файл vite.config.js.

В файле vite.config.js располагающийся в корневой директории проекта, прописываем название репозитория (рисунок 6.1): «base: "/Final\_certification\_Frontend/",».



Рисунок 6.1 – Редактирование vite.config.js

Затем командой npm run build, создаём пакет dist, который содержит скомпилированный код, готовый к развертыванию в браузере.

Использование команды npm run build позволяет:

* Преобразовать код TypeScript или JSX в чистый JavaScript;
* Объединить несколько модулей в один файл JavaScript;
* Удаляет ненужные пробелы, комментарии и другие символы из кода для уменьшения его размера;
* Применяет различные методы для улучшения производительности и уменьшения размера пакета;

После, содержимое dist копируем в репозиторий: <https://github.com/AnvarMGN/Final_certification_Frontend>.

Далее в настройках репозитория Settings, выбираем вкладку Pages и ставим ветку main в качестве источника публикации сайта. В течение нескольких минут генерируется url-адрес для просмотра проекта: <https://anvarmgn.github.io/Final_certification_Frontend/> .

* 1. Аудит проекта с использованием Lighthouse

В браузере Opera открываем страницу сайта, зажимаем комбинацию клавиш «Ctrl + Shift + j», открывается меню браузерных инструментов разработчика.

Выбираем вкладку Lighthouse и запускаем проверку. Результаты отображены на рисунке 7.1.

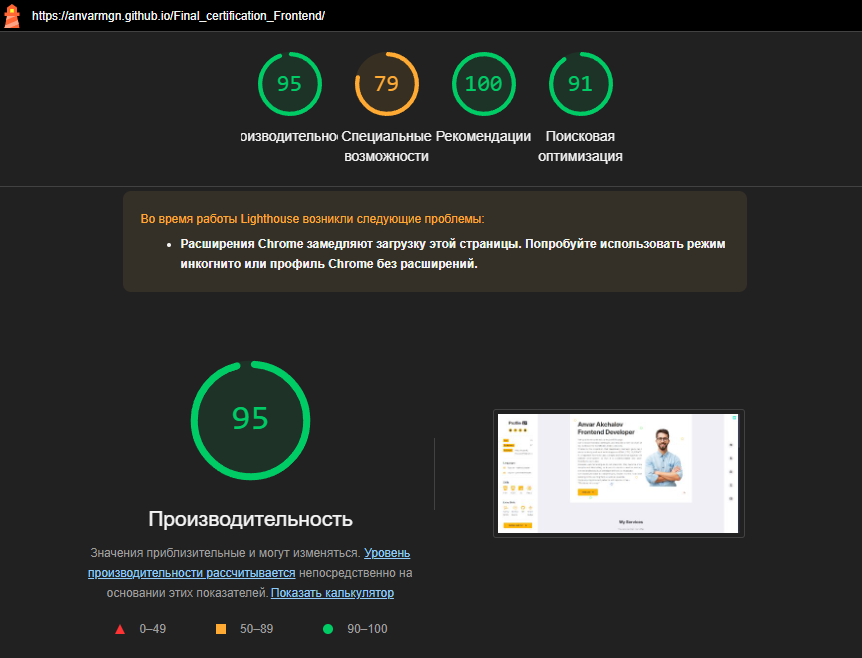


Рисунок 7.1 – Результат аудита Lighthouse

Почти все показатели находятся в зеленой зоне – это хороший показатель.

Для улучшения показателя «Специальных возможностей» предлагается:

* Сделать более контрастным цвет фона и переднего плана, таким образом улучшив читаемость текста;
* Для элементов ввода формы присвоить соответствующие ярлыки, таким образом программы специальных возможностей будут правильно озвучивать элементы управления формой;
* Увеличить текст ссылок, располагающихся в блоке Profile, для улучшения читаемости.

ЗАКЛЮЧЕНИЕ

В рамках данного дипломного проекта был проведён обзор существующих сайтов-портфолио, выделены ключевые моменты, на основании которых подобран макет Figma и спланирована структура сайта.

В разделе исследования предметной области описаны и отмечены преимущества инструментов и технологий, использованных при разработке проекта.

Так же описаны детали разработки фронтенд части веб-приложения с использование фреймворка React: файловая структура проекта, архитектура компонентов и их функционал.

Данный проект позволил закрепить изученный ранее материал и так же получить опыт в комплексном подходе разработки фронтенд части веб-приложения.

Сайт-портфолио имеет возможность дальнейшего масштабирования, добавления новой функциональности:

* К примеру, данные, которые хранит Redux store, возможно связать с телеграмм ботом, который в свою очередь будет пересылать их на e-mail или аккаунт телеграмм. Проработать валидацию формы и безопасность данных.
* Добавление новых разделов, таких как «Карьера», «Блог», «Рекомендации».
* Добавление режима «Dark mode», однако, в этом случае следует использовать более современный – модульных подход стилизации. React как раз располагает подобными библиотеками.

По мере получения практического опыта, а также изучая новые технологии и направления, коллекцию портфолио важно пополнять новыми проектами.

Сайт-портфолио демонстрирует навыки и опыт работы разработчика – цель достигнута.

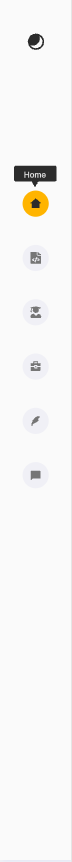
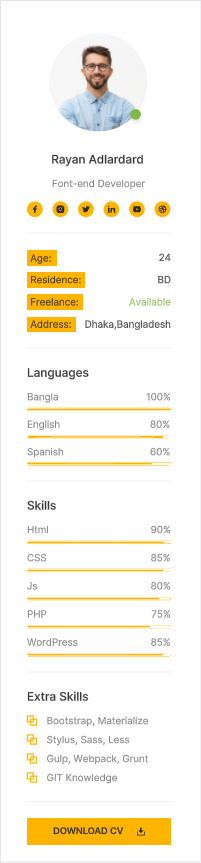
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. БЭМ // Электронная версия на сайте <https://blog.skillfactory.ru/glossary/bem/>
2. Методические материалы и презентации полученные в процессе обучения в Geekbrains за 2022-2024 года
3. Основы Sass // Электронная версия на сайте <https://sass-scss.ru/guide/>
4. Собираем портфолио начинающему веб-разработчику // Электронная версия на сайте <https://proglib.io/p/sobiraem-portfolio-nachinayushchemu-veb-razrabotchiku-2020-10-24#comments>
5. Учимся верстать: что такое CSS // Электронная версия на сайте <https://skillbox.ru/media/code/chto_takoe_css/>
6. Что я узнал, просмотрев более 40 портфолио разработчиков - 9 советов по созданию лучшего портфолио // Электронная версия на сайте <https://dev.to/kethmars/what-i-learned-after-reviewing-over-40-developer-portfolios-9-tips-for-a-better-portfolio-4me7>
7. Что такое Git // Электронная версия на сайте <https://skillbox.ru/media/code/chto_takoe_git_obyasnyaem_na_skhemakh/#stk-2>
8. Что такое JavaScript и зачем он нужен // Электронная версия на сайте

<https://skillbox.ru/media/code/chto-takoe-javascript-i-zachem-on-nuzhen/>

1. Язык HTML: что это такое и как он работает // Электронная версия на сайте <https://skillbox.ru/media/code/chto_takoe_html/>
2. JavaScript // Электронная версия на сайте <https://ru.wikipedia.org/wiki/JavaScript>
3. Pixso как инструмент для преподавания веб-дизайна // Электронная версия на сайте <https://www.pedopyt.ru/categories/19/articles/4710>
4. React // Электронная версия на сайте <https://blog.skillfactory.ru/glossary/react/>
5. Sass // Электронная версия на сайте <https://blog.skillfactory.ru/glossary/sass/>
6. Visual Studio Code // Электронная версия на сайте <https://blog.skillfactory.ru/glossary/visual-studio-code/>
7. Vite – Frontend инструмент нового поколения // Электронная версия на сайте <https://vite-docs-ru.vercel.app/>
8. 7 Лучших библиотек управления состоянием React // Электронная версия на сайте <https://trio.dev/7-top-react-state-management-libraries/>

Приложение А

(обязательное)



