Введение

Web-сайты. Без них невозможно представить существование в 2019 году. Создано огромное количество Web-сайтов на любую тематику, на которых находится почти вся известная человечеству информация. С помощью Web-сайтов любой человек может каждый день заниматься саморазвитием, учиться, работать или развлекаться.

Развлекательных сайтов больше, наверное, чем рыб в океане. И на них можно смотреть разные видео и фильмы, сериалы, слушать музыку в хорошем качестве, знакомиться и общаться с новыми людьми на форумах или имиджбордах, играть в браузерные игры, читать журналы, новости и книги в электронном виде, изучать разнообразные предложения магазинов одежды или электроники и всегда быть в курсе новинок интересующей вас индустрии.

Используя Web-сайты можно также решать бытовые задачи: узнать рецепт борща, если ваша бабушка уже не в состоянии вам помочь, узнать сколько можно пить таблеток, которые вы нашли на улице, спросить совета о том, как завести разговор на интересующую вас тему с каким либо человеком или даже как провести операцию на сердце, если вас устроили на должность хирурга “по блату”.

Исходя из того, для какой сферы разрабатывали сайт, и их реализации их делят на определённые категории. Ниже обозначены основные тематики Web-сайтов. Это лишь наиболее распространенные из них, на деле их существует значительно больше.

Покупки онлайн. Такие сайты, на которых можно заказать еду, одежду, электронику, билеты в кино или театр, дешёвый китайский ширпотреб, который будет доставляться несколько месяцев и многое многое другое.

Имиджборды и форумы. Внутри этой категории сайтов существует ещё огромное количество подкатегорий. Существуют форумы для молодых мам, пожилых мам, программистов, киберспортсменов, любителей авто, садоводов и много много других. А имиджборды это вообще интересная и очень глубокая тема. Имиджборды, это разновидность форумов, отличительные черты которых - это возможность общаться анонимно, а также достаточно “особенный” стиль общения. Также имиджборды разделены на “доски” (темы) в которых создаются “треды” (от англ. thread - “нить”). Люди могут создавать посты с темой для обсуждений (треды) на соответствующей “доске” и другие аноны будут создавать там посты и отвечать создателю треда и друг другу. На имиджбордах можно встретить абсолютно любую личность. На имиджборде может сидеть как известный политик или делящийся своим опытом успешный программист, так и травимый своими одноклассниками школьник, который пришёл пожаловаться на жизнь, и на имиджборде его выслушают и иногда даже поддержут.

Поиск работы. Можно разместить резюме, просмотреть вакансии, отправить заявки и получить уведомления — обычно такие сайты работают в связке с приложением.

Социальные сайты. Социальные сети. Они удобны для быстрого общения и обмена информацией, просмотра новостей и уведомлений. Они так же обычно работают в связке с приложением.

Новости. Газеты, журналы и другие СМИ. Такие сайты удобны и значительно расширяют аудиторию изданий.

Вывод: Сайты охватывают все сферы быта - от работы до досуга и даже личной жизни. Без Web-сайтов невозможно прожить ни одного полноценного дня в 2019-ом году.

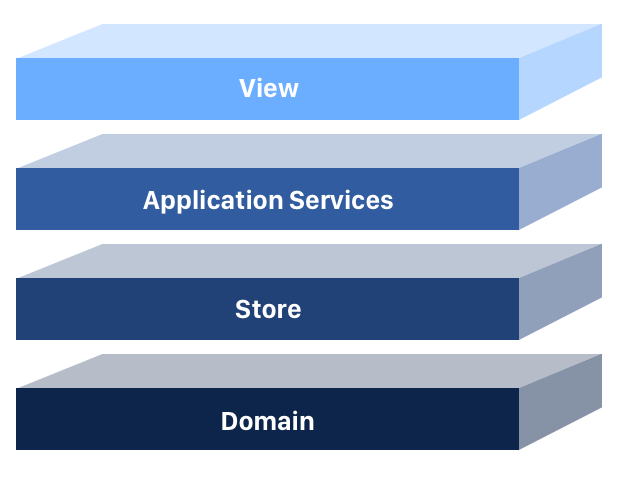
Теоретические основы

1.1 **Понятие и назначение одностраничного web-портала (SPA)**

Single Page Application – сокращенно SPA, в переводе на русский язык означает “Приложение одной страницы”. Другими словами SPA – это web-приложение, размещенное на одной web-странице, которая для обеспечения работы загружает весь необходимый код вместе с загрузкой самой страницы. Приложение такого типа появились сравнительно недавно, с началом эры HTML5 и SPA является типичным представителем приложений на HTML5.

Как мы знаем, HTML5 это нечто иное как HTML + CSS3 + JavaScript + [несколько новых тегов]. Таким образом, SPA - это приложения, написанные на языке JavaScript. И, следовательно, немного перефразировав предыдущие определения, получаем:

SPA – это web-приложение, размещенное на одной странице, которая для обеспечения работы загружает все javascript-файлы (модули, виджиты, контролы и т.д.) , а также файлы CSS вместе с загрузкой самой страницы.



*1.Четыре уровня одностраничных (SPA) приложений*

Несмотря на общепринятое мнение, центральное место в процессе дизайна и [разработки веб-сайтов](https://xbsoftware.ru/website-razrabotka/) не всегда занимает фаза написания кода. В первую очередь приходящие на ум технологии, такие как HTML, CSS и JavaScript, и в самом деле создают образ Сети, к которому мы привыкли и определяют способы нашего взаимодействия с информацией. Что обычно остается вне поля зрения, но в то же время является едва ли не самой важной частью процесса разработки, так это стадии предварительного сбора информации, тщательного планирования, а также поддержки уже после запуска сайта. Можно выделить разное количество этапов, из которых состоит процесс разработки. Обычно это число от пяти до восьми, но в каждом случае общая картина остается примерно одинаковой. Давайте остановимся на среднем значении. Итак, семь основных этапов разработки:

1) Сбор информации

2) Планирование

3) Дизайн

4) Создание контента

5) Разработка

6) Тестирование, обзор и запуск

7) Поддержка.

Основными элементами, использующимися при построении SPA, являются:

* [Фреймворки](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) для JavaScript, в частности [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller) и [MVVM](https://ru.wikipedia.org/wiki/MVVM)-фреймворки
* Роутинг: навигация между представлениями (view) производится во [фронтенде](https://ru.wikipedia.org/wiki/Front_and_back_ends)
* [Шаблонизатор](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80)
* [HTML5](https://ru.wikipedia.org/wiki/HTML5)
* [API](https://ru.wikipedia.org/wiki/API) для [бэкэнда](https://ru.wikipedia.org/wiki/Front_and_back_ends), например, в стиле [REST](https://ru.wikipedia.org/wiki/REST)
* [Ajax](https://ru.wikipedia.org/wiki/Ajax)

1.2 **Функциональные возможности одностраничного клиента, реализованного на основе JavaScript**

[JavaScript](https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/JavaScript) ("JS" для краткости) — это полноценный [динамический язык программирования](https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/Dynamic_programming_language), который применяется к [HTML](https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/HTML) документу, и может обеспечить динамическую интерактивность на веб-сайтах. Его разработал Brendan Eich, сооснователь проекта Mozilla, Mozilla Foundation и Mozilla Corporation.

JavaScript невероятно универсален. Вы можете начать с малого, с простых функций, таких как карусели, галереи изображений, изменяющиеся макеты и отклик на нажатие кнопок. Обладая большим опытом, вы сможете создавать игры, анимированную 2D и 3D графику, полномасштабные приложения с базами данных и многое другое!

JavaScript сам по себе довольно компактный, но очень гибкий. Разработчиками написано большое количество инструментов поверх основного языка JavaScript, которые разблокируют огромное количество дополнительных функций с очень небольшим усилием. К ним относятся:

* Программные интерфейсы приложения ([API](https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/API)), встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание HTML и установку CSS стилей, захват и манипуляция видеопотоком, работа с веб-камерой пользователя или генерация 3D графики и аудио сэмплов.
* Сторонние API позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков, таких как Twitter или Facebook.
* Также вы можете применить к вашему HTML сторонние фреймворки и библиотеки, что позволит вам ускорить создание сайтов и приложений.

Существует большое количество базовых библиотек (фреймворк – от английского слова framework – “основа, структура, каркас”), которые реализуют принцип Single Page Application.

Что дают эти фреймворки:

* обеспечивают базовые принципы для SPA разработки, минимизируя трудозатраты на решение универсальных задач (смотри раздел “Составляющие SPA);
* фреймворки созданы сообществом разработчиков, а значит используют опыт создания сайтов множества программистов;
* фреймворки являются отправной точкой для создания структуры на основе Single Page Application.

JavaScript является [объектно-ориентированным](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) языком, но используемое в языке [прототипирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих [функциональным языкам](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — функции как [объекты первого класса](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0), объекты как списки, [карринг](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D1%80%D0%B8%D0%BD%D0%B3), [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D1%8B%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8), [замыкания](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) — что придаёт языку дополнительную гибкость.

Несмотря на схожий с С синтаксис, JavaScript по сравнению с языком С имеет коренные отличия:

* [объекты](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) с возможностью [интроспекции](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5));
* функции как [объекты первого класса](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0);
* автоматическое [приведение типов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2);
* автоматическая [сборка мусора](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5));
* [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D1%8B%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8).

В языке отсутствуют такие полезные вещи, как:

* [стандартная библиотека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0): в частности, отсутствует [интерфейс программирования приложений](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) по работе с файловой системой, управлению потоками ввода-вывода, базовых типов для [бинарных данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB);
* стандартные [интерфейсы](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) к [веб-серверам](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) и [базам данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85);
* [система управления пакетами](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D0%B0%D0%BC%D0%B8), которая бы отслеживала зависимости и автоматически устанавливала их.

1.3 **Достоинства и недостатки применения SPA-приложений**

### Преимущества одностраничных приложений:

SPA характеризуются отличным быстродействием, так как большинство ресурсов, которые они используют (HTML+CSS+Скрипты), загружаются лишь однажды в течение сессии использования приложения. После совершения действий на странице меняются лишь данные.

Богатый пользовательский интерфейс, так называемый User Experience. Так как web-страница одна, построить богатый, насыщенный пользовательский интерфейс гораздо проще. Проще хранить информацию о сеансе, управлять состояниями представлений (views) и управлять анимацией (в некоторых случаях).

Разработка веб-приложений обычно быстрее и эффективнее. Нет необходимости писать отдельный код для рендера страницы на стороне сервера. Также гораздо легче запустить процесс разработки подобных приложений, потому что писать код можно начинать с файла file://URI, не используя при этом никакой сервер.

SPA существенно (в разы) сокращает так называемые “хождения по кругу”, то есть загрузку одного и того же контента снова и снова. Если ваш портал (сайт) использует шаблон, то вместе с основным содержанием какой-либо страницы посетитель сайта обязательно загружает разметку шаблона. Да, кэширование данных на данном этапе развития WWW достигло высочайших результатов, но если нечего кэшировать, то и время, и ресурсы на это не тратятся.

SPA оптимизированы для Chrome debugging, разработчики могут отслеживать сетевые действия, изучать элементы страниц и данные, с ними ассоциируемые.

Если у вас уже есть SPA, будет возможность с тем же бэкендом создать и мобильное приложение.

SPA более эффективны в кэшировании данных на локальных носителях. Приложение высылает один запрос, собирает все необходимые данные, и с этого момента способно работать даже в режиме оффлайн.

### Недостатки одностраничных приложений:

SEO-оптимизация одностраничных приложений, по очевидным причинам, не очень проста. Контент приложений загружается при помощи AJAX (Asynchronous JavaScript and XML) — метода обмена данными и обновления приложения без перезагрузки страницы, в то время как SEO-оптимизация основана на устойчивости контента в каждой отдельно взятой странице.При этом продвижение вашего сайта в поисковиках не невозможно. Многие изменения в SEO можно провести на стороне сервера, а компании вроде Google продолжают придумывать новые решения для того, чтобы облегчить жизнь как владельцам SPA, так и их пользователям.

Они довольно долго загружаются, поскольку тяжелые клиентские фреймворки должны сперва загрузиться в браузер.

SPA требуют JavaScript в активном режиме в браузерах пользователей. Если кто-то из ваших клиентов вручную отключит использование JavaScript, они не смогут в полной мере воспользоваться вашим приложением. Даже если вы будете кэшировать и обрабатывать ваши страницы на стороне сервера (а это сейчас тоже возможно), вы всё ещё рискуете не доставить пользователям без JS все функции одностраничного приложения в правильном виде.

По сравнению с традиционными приложениями, SPA чуть хуже защищены. Благодаря межсайтовому скриптингу (XSS), злоумышленники имеют возможность внедрять дополнительные скрипты на стороне клиента.

Некоторые утечки памяти в JavaScript могут привести к падению производительности даже в мощных системах

1.4

Практическая часть

2.1 **Постановка задачи**

Целью курсового проектирования является приобретение практических навыков по проектированию и программированию на JavaScript, а также освоение базовых и профессиональных компетенций: способствует формированию базовых ОК (1-10)

ПК 1.2. Осуществлять выбор технологии, инструментальных средств и средств ВТ при организации процесса разработки и исследования объектов профессиональной деятельности.

ПК 1.3. Обеспечивать защиту информации в сети с использованием

программно-аппаратных средств.

ПК 1.4. Принимать участие в приемо-сдаточных испытаниях

компьютерных сетей и сетевого оборудования различного уровня и в оценке качества и экономической эффективности сетевой топологии.

ПК 1.5. Контролировать соответствие разрабатываемых проектов и

технической документации стандартам, техническим условиям и другим

нормативным документам.

2.2 **Реализация интерфейса одностраничного web-клиента**

Интерфейс включает комплекс средств и методов взаимодействия между элементами системы. Оформление сайта обеспечивает взаимодействие клиента и фирмы, реализующей определенный проект.

Приятный и удобный стиль сайта и его интерфейс привлекают пользователей и облегчают знакомство и пользование сайтом. Правильное расположение сервисов и ссылок увеличивает эффективность пользования ресурсом на подсознательном уровне.

Вышеперечисленные факторы относят разработку программы оформления веб-страниц к первоочередным задачам в реализации своего проекта.

Такую важную часть на пути к реализации своего проекта не стоит доверять дилетантам, ведь в первую очередь, страница сайта – это программное обеспечение с грамотно выраженными мыслями, призывами посредством контента. По итогу разработчики и программисты должны достигнуть такого уровня, чтоб пользователь, который впервые посетил ресурс, начал на интуитивном уровне переходить по внутренним и внешним ссылкам и инструментам.

Этапы проектирования:

* Ознакомление и анализ аудитории конечного потребления. Когда имеются усредненные данные о потенциальном пользователе, тогда создать более эргономичные условия и оформление.
* Создание макета функционала. Требуется осмыслить и определиться с возможностями страницы для полной реализации задумки.
* Специалист по проектированию должен представлять, и понимать какой будет результат, то есть действовать согласно намеченной концепции.
* Создание структуры и оформление. На данном этапе проявляют себя дизайнеры, которые ведут проведение работ к окончанию.
* Пробный запуск и тестирование работы. Данная процедура позволяет оценить организованную работу, а так же выявить и устранить ошибки и недочеты. По окончании данного процесса сайт загружается в сеть «Всемирной паутины».

Свой проект я начал реализовывать с постановки задач, выбора темы, поиска информации.

Затем я создал папку, поместив туда три файла с расширением .html, .css, .js.

Я начал писать в VSCode, потому что он бесплатный, я пишу в нём уже долгое время, я к нему привык, а также он имеет ряд преимуществ:

• Visual Studio Code поддерживает работу с TypeSсript, JavaSсript, Node.js и Mono.  
• Имеются встроенные отладчик и командная строка.  
• Поддержка практически всех языков программирования.  
• Наличие встроенной библиотеки элементов кода.  
• Автозавершение при вводе кода.  
• Добавление в библиотеку собственных сниппетов.  
• Подсветка синтаксиса.  
• Одновременная работы с несколькими проектами.  
• Поддержка многооконного и двухпанельного режимов.  
• Расширение функционала с помощью плагинов.  
• Интеграция с Visual Studio Team Services, GitHub и GIT.  
• Наличие встроенных средств для тестирования, сборки, упаковки и развертывания приложений.  
• Публикация созданных программных продуктов в Microsoft Azure (через посредство Visual Studio Team Services).  
• Интегрированная система подсказок.  
• Командная работа над проектами.  
• Широкий набор настроек и кроссплатформенность.

Начал я с того, что наверстал свой шаблон, а именно создал нужные мне блоки с кнопками и разные окна, которые будут нужны в ходе работы с сайтом. Первый этап был самым простым, но он достаточно быстро закончилсяи мне пришлось заниматься вторым этапом:

Этап оформление всего вышесказанного в CSS. Этот этап оказался тоже не сложным, мне было приятно раскрашивать свои “дивы” и “баттоны” в нужные мне цвета.

Самый главный этап создания моей работы – это функции на языке программирования JavaScript. В моей работе я использовал функции для открытия модальных окон, сохранение списка заказов, рендера заказов, и куча обработчиков событий для обработки кликов по кнопкам, такие как скрытие некоторых элементов и открытие других элементов на их месте.

2.3 **Реализация на скриптов на основе JavaScript**

* Рендер списка заказов

|  |
| --- |
| const renderOrders = () => { |
|  |  |
|  | ordersTable.textContent = ''; |
|  |  |
|  | orders.forEach((order, i) => { |
|  | console.log(order); |
|  | ordersTable.innerHTML += ` |
|  | <tr class="order ${order.active ? 'taken' : ''}" |
|  | data-number-order="${i}"> |
|  | <td>${i + 1}</td> |
|  | <td>${order.title}</td> |
|  | <td class= ${order.currency}></td> |
|  | <td>${order.deadline}</td> |
|  | </tr>`; |
|  | }); |
|  | }; |

* Открытие модального окна

|  |
| --- |
| const openModal = (numberOrder) => { |
|  | const order = orders[numberOrder]; |
|  |  |
|  | const { title, firstName, email, phone, description, |
|  | amount, currency, deadline, active = false } = order; |
|  |  |
|  | const modal = active ? modalOrderActive : modalOrder; |
|  |  |
|  | const firstNameBlock = modal.querySelector('.firstName'), |
|  | titleBlock = modal.querySelector('.modal-title'), |
|  | emailBlock = modal.querySelector('.email'), |
|  | descriptionBlock = modal.querySelector('.description'), |
|  | deadlineBlock = modal.querySelector('.deadline'), |
|  | currencyBlock = modal.querySelector('.currency\_img'), |
|  | countBlock = modal.querySelector('.count'), |
|  | phoneBlock = modal.querySelector('.phone'); |
|  |  |
|  |  |
|  | modal.id = numberOrder; |
|  | titleBlock.textContent = title; |
|  | firstNameBlock.textContent = firstName; |
|  | emailBlock.textContent = email; |
|  | emailBlock.href = 'mailto:' + email; |
|  | descriptionBlock.textContent = description; |
|  | deadlineBlock.textContent = deadline; |
|  | currencyBlock.className = 'currency\_img'; |
|  | currencyBlock.classList.add(currency); |
|  | countBlock.textContent = amount; |
|  | phoneBlock ? phoneBlock.href = 'tel:' + phone : ' '; |
|  |  |
|  | modal.style.display = 'flex'; |
|  |  |
|  | modal.addEventListener('click', handlerModal); |
|  | }; |

* Скрипт внутри модального окна. Закрытие, принятие заказа, отмена заказа, и готовность заказа.

|  |
| --- |
|  |
| const handlerModal = (event) => { |
|  | const target = event.target; |
|  | const modal = target.closest('.order-modal'); |
|  | const order = orders[modal.id]; |
|  |  |
|  | const baseAction = () => { |
|  | modal.style.display = 'none'; |
|  | toStorage(); |
|  | renderOrders(); |
|  | } |
|  |  |
|  | if (target.closest('.close') || target === modal ) { |
|  | modal.style.display = 'none'; |
|  | } |
|  |  |
|  | if (target.classList.contains('get-order')) { |
|  | order.active = true; |
|  | baseAction(); |
|  | } |
|  | if (target.id === 'capitulation') { |
|  | order.active = false; |
|  | baseAction(); |
|  | } |
|  | if (target.id === 'ready') { |
|  | orders.splice(orders.indexOf(order), 1); |
|  | baseAction(); |
|  | } |
|  | } |
|  |  |

Заключение

По итогу проделанной работы я приобрёл практические знания по программированию на языке JavaScript. Создав этот одностраничный сайт я изучил много полезной в работе информации а также получил отличный опыт написания скриптов на этом языке программирования.

Список литературы

* https://ru.wikipedia.org/wiki/Имиджборд
* https://learn.javascript.ru/task/modal-dialog
* <https://ruseller.com/lessons.php?rub=2&id=2810>
* <https://codepen.io/jjmartucci/pen/DnFfl>
* <https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/JavaScript_basics>
* <http://www.codenet.ru/webmast/js/spa/>
* <https://habr.com/ru/company/mailru/blog/351710/>
* <https://ozitag.com/ru/blog/spa-advantages/>
* <https://learn.javascript.ru/dispatch-events>
* https://2ch.hk/b/