Geekbrains

# Изучение освоения навыков веб-разработки на примере создания веб-сайта школы программирования по опыту обучения в ООО “ГикБреинс”

Программа: Цифровые профессии

Специализация: Frontend - программист

Семёнов И. В.

Красноярск

2024

# Оглавление

# 1. Введение

**Тема проекта:** Изучение освоения навыков веб-разработки на примере создания одностраничного веб-сайта для условной школы программирования по опыту обучения в компании ООО «ГикБрейнс».

**Цель:** Изучить особенности освоения профессии IT-сферы с нуля на примере создания клиентской части веб-сайта в условиях отсутствия опыта веб-разработки и скорректировать подход к дальнейшему изучению профессии на основе возникших сложностей.

**Какую проблему решает**: Для начинающего разработчика, при подходе к освоению новой, сложной профессии, в условиях, когда предмет изучения находится в постоянном изменении и развитии, данное исследование предназначено выработать важные ориентиры, необходимые для уверенного продвижения к намеченной цели.

**Задачи:**

1. Изучить оценку целесообразность перехода на новую профессию с точки зрения её перспективности.
2. Оценить значимость выбранной специальности для общества.
3. Изучив имеющийся макет, выделить основные элементы макета и оптимальные способы их создания в вёрстке.
4. Изучить минимальный набор инструментов, необходимый для выполнения первого проекта.
5. Наметить основные методики, необходимые для получения результата.
6. Выполнить вёрстку сайта в соответствии с макетом, создать минимально необходимую его интерактивность.
7. Выявить и отметить основные трудности, которые могут возникнуть при выполнении этой работы.
8. Сформулировать рекомендации начинающему frontend – программисту для преодоления подобных трудностей в дальнейшем.

## 1.1. Тенденции возрастания спроса на специалистов IT-сферы

Ключевым фактором развития современного общества и экономики является цифровизация всех сфер деятельности человека. Цифровизация все глубже проникает во все аспекты производственной, финансовой, социальной и государственной деятельности и является сегодня неотъемлемым условием развития страны и повышения качества жизни ее граждан.

Для сохранения конкурентоспособности экономики страны на международной арене в России разработана и реализуется национальная программа «Цифровая экономика Российской Федерации» [1], включающая федеральный проект «Кадры для цифровой

экономики», направленный на обеспечение подготовки высококвалифицированных IT-специалистов.

В связи с тем, что на современном рынке труда существует серьезный дефицит IT-специалистов, необходимых для успешного обеспечения развития цифровой экономики [2], значимым мероприятием проекта «Кадры для цифровой экономики» является ежегодное увеличение контрольных цифр приема в российские вузы на информационно-технологические направления подготовки. К 2024 г. этот показатель должен составить 120тыс. (для сравнения – 50 тыс. бюджетных мест в 2018 г.) [3].

Таким образом, перед российскими вузами стоит серьезная задача подготовки конкурентоспособных на рынке труда работников IT-сферы. Однако сегодня профессии IT-специалистов входят в топ наиболее высокооплачиваемых, а область развития и использования информационных технологий является одной из самых стабильных на рынке труда. Результаты совместных исследований рынка IT, проведенных компаниями Яндекс и HeadHunter [4] подтверждают рост спроса на IT-специалистов (табл. 1).

*Таблица 1*

**Рейтинг изменений востребованности IT-специалистов на российском рынке труда**

| Вакансия | Изменение потребностей за два года, в % |
|---|---|
| FrontEnd-разработчик (разработчик, создающий видимую для пользователя часть веб-страницы) | +20 |
| PHP-разработчик | –21 |
| Java-разработчик | –13 |
| Системный администратор | +1 |
| Тестировщик | 0 |
| 1С-разработчик | +17 |
| .NET-разработчик | –4 |
| C++-разработчик | +2 |
| iOS-разработчик | –17 |
| Android-разработчик | –3 |
| Python-разработчик | +21 |
| DevOps-инженер (специалист по согласованию разработки и эксплуатации кода) | +70 |
| JavaScript-разработчик | –6 |
| Data Scientist (анализ данных) | 226 |
| FullStack-разработчик (разработчик, способный в одиночку заменить команду в малом проекте) | 199 |

Таким образом, перед российскими вузами стоит серьезная задача подготовки конкурентоспособных на рынке труда работников IT-сферы. Однако сегодня профессии IT-специалистов входят в топ наиболее высокооплачиваемых, а область развития и

использования информационных технологий является одной из самых стабильных на рынке труда. Результаты совместных исследований рынка IT, проведенных компаниями Яндекс и HeadHunter [4] подтверждают рост спроса на IT-специалистов (табл. 1).

Высокий спрос на профессионалов в области web-разработки подтверждается данными, представленными на сайтах ведущих рекрутинговых компаний [5; 6]. А как следует из табл. 1, лидеры спроса со стороны работодателей – специалисты в области web-разработки.

Востребованность специалистов данного профиля имеет место и на международном рынке труда. Так, в документе Information Technology Curricula 2017 (ITC-2017) [7] приведены рейтинги изменений востребованности IT-специалистов различных профессий (табл. 2).

*Таблица 2*

**Рейтинги изменений востребованности IT-профессий на международном рынке труда**

| Вакансии | Общее количество | Динамика изменений, в % | |
|---|---|---|---|
| | | Работода-тели | Представите-ли отрасли |
| Web-разработчик | 148 500 | 27 | 39 |
| Аналитик компьютерных систем | 567 800 | 21 | 33 |
| Разработчик программного обеспечения бизнес-при-ложений | 718 400 | 19 | 31 |
| Аналитик в области информационной безопасности | 82 900 | 18 | 36 |
| Разработчик программного обеспечения системных приложений | 395 600 | 13 | 31 |
| Специалист технической поддержки пользователей | 766 900 | 12 | 31 |
| Администратор баз данных | 120 000 | 11 | 26 |
| Исследователь в области Computer Science | 25 600 | 11 | 21 |
| Архитектура компьютерных сетей | 146 200 | 9 | 21 |
| Системный администратор | 382 600 | 8 | 31 |

В табл. 2 представлен прогнозируемый рост IT-профессий на 2014–2024 гг. во всех секторах деятельности (прогноз занятости), кроме того, приведено количество вакансий по этим же профессиям в соответствии с данными за 2014 г. Обращает на себя внимание значительное изменение позиции в рейтинге востребованности профессии webразработчика (Web Developers): в 2014 г. Web Developers занимали седьмую позицию по количеству имеющихся вакансий, а к 2024 г. прогнозируется, что данные специалисты займут первую строку рейтинга [7]. Профессии web-разработчиков имеют самый большой прогнозируемый рост в IT-индустрии – 39%.

Исходя из возрастающего спроса на профессионалов в области web-разработки,

ведущие российские вузы и IT-компании расширяют спектр своих услуг для подготовки востребованных специалистов в этой области, разрабатывая программы переподготовки и курсов повышения квалификации, в частности, на основе учебных онлайн-курсов [15; 16].

В настоящее время получение образования в формате онлайн приобретает все большую популярность. В мире существуют различные мнения об эффективности онлайнкурсов по сравнению с традиционным обучением [17]. В России, согласно исследованию П. Л. Пеккер, имеются широкие потенциальные возможности успешного использования онлайн-курсов, которые обусловлены прежде всего протяженностью территории и численностью населения страны [18]. В своих исследованиях компания HeadHunter отмечает, что одной из популярных категорий онлайн-курсов являются курсы по IT-направлениям: им отдало предпочтение около 23% респондентов [19].

При этом, по результатам исследования компании EdTech, проведенного в 2019 г., в качестве основной цели обучения большинство респондентов указывают «обучиться отдельным практическим навыкам в своей профессии», то есть углубить практикоориентированную составляющую и при этом развить так называемые soft skills [20].

В соответствии с запросами современного общества многие ведущие вузы, центры профессиональной переподготовки, а также крупные компании разрабатывают и реализуют онлайн-курсы по программированию, ориентированные как на новичков в области информационных технологий, так и на лиц, имеющих базовое образование по данному направлению.

Наиболее популярными являются онлайн-курсы, направленные на подготовку слушателей в области web-программирования. Так, курсы по данному направлению предлагают Mail.ru Group, GeekBrains, Яндекс Практикум, Нетология-групп, SkillBox, HTML Academy и др.

### 1.2. Роль веб-приложений в цифровизации общества.

В настоящее время особую роль в развитии экономики имеют процессы внедрения современных цифровых технологий, автоматизации рутинных задач, которые приходилось вручную. Немалый вклад в этот процесс внесли веб приложения, появившиеся в процессе развития глобальной паутины. Теперь же все этапы работы многих специалистов разных сфер, развлечения и многое другое может производиться онлайн. Достигается это посредством устройства веб-приложений − основная вычислительная часть работы производится на сервере, когда пользователь получает готовый результат у себя на

устройстве В 2024 году мало какой человек может себе представить жизнь без социальных сетей, мессенджеров и YouTube. Всё это в основе своей – веб-приложения. Но так было не всегда. Ещё всего лишь 30 лет назад открываемые браузером сайты могли похвастаться только статичной картинкой, что позволяло лишь реализовать сайт-визитку.

В 1991 году британский ученый Тим Бернерс-Ли разработал первый веб-сервер и клиент, создав основу для развития веб-разработки. Он создал язык разметки гипертекста, известный как HTML, и первый браузер под названием WorldWideWeb. Вместе с этим, Бернерс-Ли также разработал URL (унифицированные указатели ресурсов) и HTTP (протокол передачи гипертекста), что позволило установить стандартную схему работы веб-страниц.

Середина 1990-х годов стала переломным моментом для веб-разработки, когда введение CGI (Common Gateway Interface) дало возможность разработчикам использовать динамические веб-страницы и взаимодействовать с базами данных. На этом этапе, благодаря новой технологии, они уже могли создавать интерактивные сайты и приложения

Развитие Интернета и появление новых технологий сделали веб-разработку более сложной и разнообразной. В конце 1990-х годов был создан JavaScript – язык программирования, который позволил добавлять интерактивность и анимацию на веб-страницы. Вместе с тем, появились новые языки программирования и технологии, такие как CSS (каскадные таблицы стилей) для стилизации страниц, PHP и MySQL для работы с базами данных, а также фреймворки и библиотеки, которые упростили и ускорили процесс разработки. В 1996 появляется Macromedia Flash, давший возможность анимировать веб-страницы.

В это же время появляются первые социальные сети и простые веб-приложения, а также.меняются технические спецификации сайтов, способы проектирования и использования веб-страниц.

Интернет становится местом скопления онлайн-инструментов и платформ, на которых люди делятся своими взглядами, мнениями, мыслями и опытом. На этом этапе развития интернета в разработке сайтов используются технологии веб-браузеров, например, фреймворк AJAX. Именно фреймворки были самыми популярными среди разработчиков. Охарактеризовать этот период можно так - AJAX, оптимизация тегов, нововведения в области фронт-енда.

Следующий этап развития - это результат эволюции использования интернета и взаимодействия в его пределах. Интернет стал базой данных. Этот этап характеризуется оптимизацией внутренней части сайтов, этому уделялось большое внимание. Именно на этом

этапе развития интернета было внесено больше всего инноваций в работу сайтов и интернета в целом. Что характерно для этого периода, так это то, что данные не принадлежат конкретным организациям или людям, а совместно используются пользователями. Этот этап развития интернета также называется Semantic Web.

Интернет превратился из сайтов, доступных только для чтения, в сегодняшний многофункциональный Semantic Web. Веб приложения позволяют производить различные действия, которые ранее делались на бумаге в течение многих суток. Теперь же достаточно сделать несколько кликов и всё готово. Совсем скоро этап развития интернета сменится новым, обеспечивающим прочную связь между человеком и машиной для создания интерактивных данных. Блокчейн-проекты, такие как Free TON, ChainLink, Polkadot способствуют более быстрому развитию интернета. Уже сейчас микросервисы и облачные хранилища позволяют хранить огромное количество данных и получить к ним доступ при наличии интернета. Теперь же, новые технологии, такие как криптовалюты и блокчейн, принесли очень много новшеств в интернет, например, децентрализованные финансы (DeFi), децентрализованные приложения (dApps), невзаимозаменяемые токены (NFT).

Сегодня веб-разработка продолжает развиваться со взрывным ростом мобильных устройств и расширением возможностей сети Интернет. Веб-разработчики используют различные языки программирования, фреймворки, библиотеки и инструменты для создания современных и уникальных веб-сайтов и приложений. Интерактивность, анимации, адаптивный дизайн – все это стало обычной частью современной веб-разработки, а будущее сулит еще больше инноваций и технологических прорывов.

Материалы, кратко изложенные в этом разделе, приводят к мысли о безусловной целесообразности войти в сообщество участников процессов, происходящих в наше время в сфере компьютерных технологий и, особенно, в сфере веб-разработки. В настоящий момент наиболее доступным для таких начинаний является frontend-разработка, изучение которой на примере создания сайта и является темой данного проекта.

# 2. Основная часть

## 2.1. Краткий обзор этапов создания сайта.

Очевидно, что в реальной практике постановка конечной цели и образ требуемого результата является прерогативой заказчика, который имеет конкретный предмет — как правило, свой бизнес и цель — улучшить его функционирование. Для достижения этой, как и любой другой, цели, заказчик формулирует техническое задание. После этого, как правило,привлекается подрядчик (или подрядчики) для определения этапов достижения конечной цели, а также задачи этих этапов.

Первый этап — это этап предварительного исследования и сбора информации, который определяет то, как будут протекать все последующие стадии разработки. Самое важное на этом этапе — получить ясное и полное понимание того, каким будет назначение будущего сайта, каких целей заказчик хочет достичь с его помощью, а также какова целевая аудитория, которую планируется на него привлечь. Такая своеобразная анкета веб-разработки позволит определить наилучшую стратегию дальнейшего развития проекта. Новостные порталы отличаются от развлекательных сайтов, а сайты для подростков отличаются от таковых для взрослой аудитории. Разные сайты предоставляют посетителям разную функциональность, а значит разные технологии должны использоваться в том или ином случае. Детально составленный план, созданный на основе данных, полученных на этом этапе, может предотвратить от затраты дополнительных ресурсов на решение непредвиденных трудностей, таких как смена дизайна или добавление функционала, не предусмотренного изначально.

Второй этап — это планирование: создание карты сайта и макета. На этой стадии разработки заказчик уже может получить представление о том, каким будет будущий сайт. На основе информации, собранной на предыдущей стадии, создается карта сайта (sitemap). Карта сайта описывает взаимосвязь между различными частями сайта. Это помогает понять, насколько удобным в использовании он будет. По карте сайта можно определить «расстояние» от главной страницы до других страниц, что помогает судить о том, насколько просто пользователю будет добраться до интересующей его информации. Основная цель создания карты сайта — создать легкий с точки зрения навигации и дружественный к пользователю продукт. Это позволяет понять внутреннюю структуру будущего сайта, но не описывает то, как сайт будет выглядеть.

Иногда, прежде чем приступить к написанию кода или к разработке дизайна, может быть важным получить одобрение заказчика. В этом случае создается макет (wireframe или mock-

up). Макет представляет из себя визуальное представление будущего интерфейса сайта. Но, в отличие например, от шаблона, он не содержит элементов дизайна, таких как цвет, логотипы, и т.п. Он только описывает, какие элементы будут помещены на страницу и как они будут расположены. Макет представляет собой своего рода набросок будущего сайта.

Третий этап — это дизайн. На этом этапе веб-сайт становится еще ближе к своей окончательной форме. Весь визуальный контент, такой как изображения, фото и видео, создается именно сейчас. И опять-таки вся информация, которая была собрана на самой первой стадии проекта, крайне важна на этом шагу. Интересы заказчика, а также целевая аудитория должны учитываться в первую очередь во время работы над дизайном. Дизайнером на данном этапе создается шаблон страницы (page layout). Основное назначение шаблона — визуализировать структуру страницы, ее содержимое, а также отобразить основной функционал. На этот раз, в отличие от макета, используются элементы дизайна. Шаблон содержит цвета, логотипы и изображения. Он дает возможность судить о том, как в конечном результате будет выглядеть готовый сайт. После создания шаблон может быть отправлен заказчику. После обзора заказчиком проделанной работы, он присылает свой отзыв. Если его не устраивают какие-то аспекты дизайна, подрядчик должен изменить существующий шаблон и снова отправить его заказчику. Это должно повторяться до тех пор, пока заказчик не будет полностью удовлетворен результатом.

Четвёртым этапом является создание контента. Здесь нумерация этапав наиболее «размыта», так как контент, как правило, уже имеется в некотором количестве у заказчика. Можно сказать, что процесс создания контента обычно проходит параллельно с другими стадиями разработки. Важно, чтобы весь контент был подготовлен к моменту его размещения и использования в общем процессе создания сайта.

Пятым этапом, который, собственно и представлен в практической части, является вёрстка и разработка. Все графические элементы, разработанные ранее, используются на данной стадии. Обычно в первую очередь создается домашняя страница, а затем к ней добавляются остальные страницы в соответствии с иерархией, разработанной на этапе создания карты сайта. Также на этом этапе происходит установка CMS. Все статичные элементы веб-сайта, дизайн которых был разработан ранее при создании шаблона, превращаются в реальные динамические интерактивные элементы веб-страницы. Немаловажная задача — проведение SEO-оптимизации (Search Engine Optimization), которая представляет собой оптимизацию элементов веб-страницы (заголовков, описания, ключевых слов) с целью поднятия позиций сайта в результатах выдачи поисковых систем. Валидность кода является крайне важной в этом случае.

Следующий, шестой этап, это тестирование и запуск. Тестирование является, наверное, самой рутинной частью разработки. Каждая ссылка должна быть проверена, каждая форма и каждый скрипт должны быть протестированы. Текст должен быть проверен программой проверки орфографии для выявления возможных опечаток и ошибок. Валидаторы кода используются для того, чтобы быть уверенным, что созданный на предыдущем этапе код полностью соответствует современным веб-стандартам. Это может оказаться крайне важным, если для вас критична, например, кроссбраузерная совместимость. После всех проверок и перепроверок, сайт может быть загружен на сервер. Обычно для этого используется FTP-клиент. После загрузки сайта на сервер, необходимо провести еще один тест для того, чтобы быть уверенным, что во время загрузки не произошло непредвиденных ошибок и все файлы целы и невредимы.

Последний, седьмой этап, это поддержка и внесение изменений. Развитие сайтостроения в наше время привело к тому, что веб-сайт становится скорее сервисом, чем завершенным продуктом, который, после «изготовления» передаётся заказчику на его дальнейшее содержание. В первую очередь, существует период, называемый в технике периодом приработки, на котором возникает наибольшее количество ошибок. Поэтому, в сфере создания софта, продукт, в нашем случае сайт, может быть, до его выхода в продакшен, размещён и доступен для использования ограниченными группами клиентов. Это хороший способ избежать многих ошибок в уже выпущенном продукте. Такой подход даёт уверенность в том, что все работает, как и было запланировано, а пользователи довольны конечным продуктом. Нужно также быть готовым быстро вносить изменения, если это будет необходимо. Поэтому, очень важной задачей является решение возникших проблем настолько быстро, насколько это возможно. Иначе, пользователи могут выбрать другой ресурс, и не будут мириться с неудобствами. Ещё одна задача на этом этапе, это регулярные обновления CMS. Это позволит снизить риск ошибок и проблем с безопасностью. Итак, рассмотрим пятый этап создания сайта — вёрстка и разработка.

## 2.2. Основные инструменты разработки сайтов.

### 2.2.1. Графические редакторы

В рамках данного дипломного проекта будет разработан сайт условной школы программирования YtYt. Целью его создание является систематизация полученных в процессе обучения в ООО «ГикБрейнс» знаний и навыков frontend – разработки. Для этого был взят бесплатный макет в телеграмм-канале «Макеты для вёрстки» по адресу

https://t.me/figma2html. Как видим по адресу канала, макет разработан в программе Figma, что и определило мой выбор. До последнего времени данная программа была бесплатной, но несколько дней назад режим Dev Mode стал платным. **Figma** является одной из самых популярных программ у дизайнеров и разработчиков. Она отличается возможностью работы в реальном времени с командой, облачным хранением проектов, удобным коллаборативным режимом. Кроме данной программы для разработки макетов используются следующие инструменты: **Adobe X** – очень популярна среди дизайнеров, выделяется интуитивным интерфейсом, возможностью создания прототипов и анимаций, интеграцией с другими продуктами Adobe. **Sketch** – имеет мощные инструменты для создания пользовательских интерфейсов и возможность работы с векторной графикой, богатая библиотека плагинов. **Webflow**, основные преимущества которой, это возможность создания дизайна и разработки сайта в одной платформе, интеграция с CMS, гибкая адаптивная верстка. Здесь приведена часть самых популярных программ, используемых дизайнерами и разработчиками для создания сайтов. Здесь не затронуты такие программы, как **Pixso**, **Avokode** и другие, так как это выходит за рамки краткого обзора таких инструментов.

### 2.2.2. Редакторы кода для создания сайтов

Инструменты для написания кода, это, по сути своей, текстовые редакторы, позволяющие человеку написать инструкцию, которая затем преобразуется в код, понятный машине. Существует большое количество редакторов кода. Для выполнения данной работы мной был выбран редактор кода **Visual Studio Code**. Эта программа разработана компанией Microsoft, имеет множество плагинов для облегчения написания кода на различных языках программирования. Программа имеет множество настроек «под пользователя», удобный интерфейс для взаимодействия с системой контраля версий Git и интернет-сервисом GitHub. При всех этих достоинствах, она бесплатна.

Из популярных редакторов кода стоит отметить **Sublime Text**. Программа достаточно долго использовалась в веб-разработке и используется до сих пор. Поддерживается основными платформами, такими как Windows, Mac, Linux, очень легка и нетребовательна к «железу». Является условно-бесплатной, что, однако, не мешает использовать её полноценно, так как отличие от платной версии лишь в постоянных напоминаниях. Имеет огромное количество плагинов. Следует отметить, что переход на VSCode с этой программы не составит труда, а разработчики Microsoft постарались, чтобы при этом сохранился привычный интерфейс. Однако, пожалуй, единственная причина использовать эту программу (не считая заядлых её приверженцев), это её «легковесность», позволяющая

использовать этот редактор на маломощных машинах.

Кроме упомянутых, следует назвать программу **Atom** от компании GitHub, а также **JetBrains WebStorm** от компании JetBrains, которая отличается большим выбором дополнительных возможностей. Её существенным недостатком для тех, кто делает первые шаги в разработке, это тот факт, что программа платная.

### 2.3. Выбор программ и методологий для выполнения проекта.

#### 2.3.1. HTML – язык гипертекстовой разметки текста

Вёрстка сайта выполнена с использованием HTML – HyperText Markup Language – язык гипертекстовой разметки текста. Не смотря на то, что в аббревиатуре использовано слово Language (язык), HTML не является языком программирования. Он был изобретён на заре становления интернета в стенах Европейского центра ядерных исследований (фр. CERN - *Conseil Européen pour la Recherche Nucléaire* ) британцем Тим Бернерс-Ли (англ. *Sir Timothy John «Tim» Berners-Lee* ), разрабатывался в период между 1986 и 1991 годами. Язык основан на HTML — элементах. Любой документ состоит из HTML-элементов и текста. Сам элемент формируется с помощью тегов — открывающим и закрывающим. На настоящий момент актуальна версия HTML 5-го поколения, которая и была использована для создания сайта.

Вне всякого сомнения, язык разметки HTML является основой любого сайта, на которую, улучшая её, «навешиваются» все остальные свойства и методы, делающие сайт красочным, информативным и «живым». Как и любой другой продукт, HTML имеет свои преимущества и недостатки. К преимуществам можно отнести то, что он подходит почти для всех браузеров, а для его использования может применяться любой текстовой редактор. HTML бесплатен для использования, у него простой и понятный синтаксис, который позволяет начинать делать простенькие страницы через пару часов изучения. С другой стороны, профессионалы продолжают использовать его на любом уровне своих умений. Поскольку продукт, написанный на HTML, это текст, его загрузка не занимает много времени, а на хостинге он занимает совсем немного места, что делает дешевле использование хостинга. Кроме вышесказанного, HTML-элементы содержат в себе семантику и работают с учётом цифровой доступности — незрячие люди тоже имеют возможность пользоваться ими. Кроме того, ошибки в написании документа не приводят к краху загрузки, а, в худшем случае, при загрузке будут искажения.

Из недостатков можно отметить необходимость внесения повторяющихся изменений на сайте вручную, например заменить на каждой странице пункты меню и навигации. То же

относится и к созданию новых страниц — структуру, даже если она повторяется, придётся создвавать вручную. Следует отметить, что очень сильно меняет эту ситуацию применение других инструментов, например фреймворков. Развитие языка неизбежно ведёт к его усложнению, поэтому, для использования появляющихся новых возможностей уже недостаточно, условно говоря, везде использовать  - существует более ста элементов HTML, которые необходимо знать, понимать для чего и где их следует использовать. Хотя, оптимист бы назвал это не недостатком, а эволюцией :-). Ну и, являясь основой, скелетом для сайта, HTML в одиночку не сможет создать полноценный сложный организм. Для этого ему, как минимум, потребуется подключить CSS и Javascript, а веб-разработчику — изучить эти инструменты. Естественно, для нашего сайта мы тоже будем их использовать. Для понимания основ веб-разработки, взглянем на них чуть подробнее.

### 2.3.2. CSS – каскадные таблицы стилей

Для стилизации гипертекста мной используется CSS3 (англ. *Cascading Style Sheets*) - каскадные таблицы стилей, которые создавались с использованием препроцессора SASS, что позволило использовать вложенность, основанную на нейминге БЭМ.

CSS - это набор шаблонов (стилей), управляющих форматированием ТЕГов в HTML документе. Концепция CSS близка к концепции стилевых файлов редактора Microsoft WORD. Именно в каскадных таблицах стилей, как и в стилевых файлах редактора WORD содержится Внешний вид документа определяется набор правил оформления и форматирования документа, который просматривается WWW-браузером или редактором WORD.

В стандартном языке HTML для присвоения какому-либо элементу определенных свойств (таких, как цвет, размер, положение на странице и т. п.) приходится каждый раз описывать эти свойства. Если на одной странице должны располагаться 10 или 110 таких элементов, ничуть не отличающихся один от другого, то нужно десять или сто десять раз вставить один и тот же HTMLкод в страницу, увеличивая размер файла и время загрузки HTML страницы.

CSS действует другим, более удобным и экономичным способом. Для присвоения какому-либо элементу определенных характеристик следует один раз описать этот элемент и определить это описание как стиль, а в дальнейшем просто указывать, что элемент, который нужно оформить соответствующим образом, должен принять свойства этого стиля.

Описание стиля можно сохранить не в тексте каждой отдельной страницы, а в отдельном файле – это позволит использовать описание стиля при оформлении любого

количества Web-страниц. Есть еще одно, связанное с этим, преимущество – возможность изменить оформление любого количества страниц, исправив лишь описание стиля в одном файле. Кроме того, CSS позволяет работать со шрифтовым оформлением страниц на гораздо более высоком уровне, чем стандартный HTML, избегая излишнего утяжеления страниц графикой.

Можно сказать, что логическая структура документа определяется элементами HTML-разметки, в то время как форма представления каждого из этих элементов задается CSS-описателем элемента. Именно так: HTML определяет тип и порядок появления элементов в документе, а CSS определяет их размер, точное расположение, цвет и прочие атрибуты внешнего вида и даже форму и поведение на странице элементов.

Существует четыре способа применения стилей.

Первый способ заключается в том, что можно добавить атрибут style любому тегу на странице и прямо там написать для него стиль. Обратите внимание на то, что при этом никаких селекторов писать не нужно, так как CSS код применится только к тому тегу, для которого он написан. Однако, этот метод имеет наивысший приоритет, что снижает гибкость изменения обозначенного таким образом стиля - с помощью других способов использования CSS не получится. А манипулировать этим стилем с помощью JS при большом количестве таких элементов, не удобно и захламляет JS-код. Да и сам HTML-код, при использовании этого способа, так же проигрывает в плане захламлённости. Единственной причиной применения этого метода, на мой взгляд, является именно защита стиля данного конкретного элемента от нежелательных случайных изменений. Пример использования:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Это заголовок тайтл</title>
    </head>
    <body>
        <p style="color: red; font-weight: bold;">
            Это абзац с текстом.
        </p>
    </body>
</html>
```

Второй способ заключается в использовании тега <style>, который обычно размещают в разделе <head>, но можно и внутри раздела <body>. Особенность работы этого метода заключается в том, что стили, определённые с его помощью, действуют только на одной

странице — на которой применён этот метод. Это приводит к тому, что, например, для footer-а сайта или любого другого, повторяющегося на нескольких страницах, блока, стили придётся прописывать на каждой странице отдельно. Пример использования:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Это заголовок тайтл </title>
        <style>
            p {
                color: red;
            }
        </style>
    </head>
    <body>
        <p>
            Это абзац с текстом.
        </p>
    </body>
</html>
```

Третий способ подключения CSS, он и самый распространённый, — это когда стили хранятся в отдельном файле с расширением .css, который подключается к файлу, или даже нескольким файлам HTML одной строкой в их разделе <head>. Например:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Это заголовок тайтл</title>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <p>
            Это абзац с текстом.
        </p>
    </body>
</html>
```

Подключив такой .css файл к нескольким страницам, имеющим один и тот же, например, footer, можно, изменив стиль элемента в этом файле, изменить представление этого footer-а на всех страницах. Это большое преимущество данного метода. Справедливо и другое: мы можем для, например, нескольких секций на данной странице, создать отдельный файл .css и подключить его также, вместе с основным .css файлом. В представленном проекте использован именно этот метод, в силу его универсальности и достаточной

гибкости.

Также, имеется ещё и четвёртый метод — импорт .css файла из-за пределов данного пакета документов. Вероятно, это хорошее решение при работе над проектом нескольких программистов, которые имеют общую базу данных, куда входит и требуемый файл стилей.

В данном проекте использован, как было сказано, третий метод, но не в чистом виде, а с использованием препроцессора SASS, работа которого заключается, кроме того, ещё и в автоматическом объединении всех файлов стилей, созданных с использованием препроцессора и имеющих расширение .scss, в один общий файл style.css. Это позволяет на этапе написания стилей для отдельных страниц, работать в отдельности со стилем каждой страницы, но к страницам подключается уже объединённый препроцессором файл style.css.

### 2.3.3. Препроцессор SASS и обоснование его применения.

Для меня препроцессоры стали очень вдохновляющим открытием. Как оказалось, поддерживать порядок в стилях можно значительно проще и удобней, чем написание кода CSS непосредственно. К тому же, он органически вписался в редактор VSCode.

SASS (англ. *Syntactically Awesome Stylesheets*) — является метаязыком, что определяет его как вспомогательный язык для описания другого языка. Впервые «увидел свет» 28 ноября 2006 года Его использование упрощает и ускоряет написание CSS-кода. Ещё говорят, что это препроцессор CSS, потому что SASS использует свой синтаксис для описания стилей, на основании которого генерируется CSS-код, уже понятный любому браузеру. Как только мы начинаем пользоваться SASS, препроцессор обрабатывает наш SASS-файл и сохраняет его как простой CSS-файл, который для любого браузера ничем не отличается от кода CSS, написанного вручную.

Преимущества использования препроцессора становятся особенно очевидны при больших размерах и количествах таблиц стилей и работа с ними становится сложней. Кроме того, что препроцессор помогает упростить работу со стилями за счёт создания у них определённого порядка, он позволяет использовать функции, свойственные языкам программирования и недоступные в самом CSS. В первую очередь можно упомянуть о переменных, об уже упоминавшейся мной ранее вложенности, миксинах, наследовании и других полезных возможностях, облегчающих работу с написанием кода CSS. Рассмотрим этот вопрос подробнее.

**Переменные**

Переменные служат для хранения информации, которую вы хотите использовать на протяжении написания всех стилей проекта. Вы можете хранить в переменных цвета, стеки

шрифтов или любые другие значения CSS, которые вы хотите использовать. Чтобы создать переменную в Sass нужно использовать символ $. Например:

SCSS:

```
$font-stack:  Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

CSS:

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

Когда Sass обрабатывается, он принимает значения, заданные нами в $font-stack и $primary-color и вставляет их в обычном CSS-файле в тех местах, где мы указывали переменные как значения. Таким образом переменные становятся мощнейшей возможностью, например, при работе с фирменными цветами, используемыми на всем сайте.

**Вложенности**

При написании HTML, можно заметить, что он имеет четкую вложенную и визуальную иерархию. С CSS это не так.

SASS позволяет нам вкладывать CSS селекторы таким же образом, как и в визуальной иерархии HTML. Но следует помнить, что чрезмерное количество вложенностей делает код менее читабельным и воспринимаемым, что считается плохой практикой.

Для иллюстрации сказанного, приведем типичный пример стилей навигации на сайте:

SCSS

```
nav {
    ul {
      margin: 0;
      padding: 0;
      list-style: none;
    }

    li { display: inline-block; }

    a {
      display: block;
```

```
        padding: 6px 12px;
        text-decoration: none;
    }
}
```

Можно заметить, что селекторы ul, li, и a являются вложенными в селектор nav. Это отличный способ сделать CSS-файл более читабельным. Когда мы сгенерируем CSS-файл, то на выходе получится что-то вроде этого:

CSS

```
nav ul  {
    margin: 0;
    padding: 0;
    list-style: none;
}
nav li {
    display: inline-block;
}
nav a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
}
```

**Фрагментирование**

Можно создавать фрагменты SCSS-файла, которые будут содержать в себе небольшие отрывки SCSS, которые можно будет использовать в других SASS-файлах. Это отличный способ сделать CSS модульным, а также облегчить его обслуживание. Фрагмент — это простой SCSS-файл, имя которого начинается с нижнего подчеркивания, например, _partial.scss. Нижнее подчеркивание в имени Sass-файла говорит компилятору о том, что это только фрагмент и он не должен компилироваться в CSS. Фрагменты SCSS подключаются при помощи директивы @import.

**Импорт**

CSS имеет возможность импорта, которая позволяет разделить CSS-файл на более мелкие фрагменты и выполнить их @import. Но тогда для каждого такого фрагмента в CSS создается еще один HTTP-запрос. SCSS берет идею импорта файлов через директиву @import, но вместо создания отдельного HTTP-запроса SCSS импортирует указанный в директиве файл в тот, где он вызывается, т.е. на выходе получается один CSS-файл, скомпилированный из нескольких фрагментов.

Например, если у нас есть два следующих фрагмента SCSS-файлов — _reset.scss и base.scss, и мы хотим импортировать _reset.scss в base.scss.

SCSS

```scss
// _reset.scss
html,
body,
ul,
ol {
   margin:  0;
   padding: 0;
}
```

```scss
// base.scss
@import 'reset';
 body {
    font: 100% Helvetica, sans-serif;
    background-color: #efefef;
}
```

Мы используем @import 'reset'; в base.scss файле. Когда мы импортируем файл, то указывать расширение .scss не обязательно.

В итоге мы получим один итоговый файл CSS, который объединил два фрагмента:

CSS

```css
html,
body,
ul,
ol {
   margin:  0;
   padding: 0;
}
body {
   font: 100% Helvetica, sans-serif;
   background-color: #efefef;
}
```

**Миксины (примеси)**

Некоторые вещи в CSS весьма утомительно писать, особенно в CSS3, где плюс ко всему зачастую требуется использовать большое количество вендорных префиксов. Миксины позволяют создавать группы деклараций CSS, которые приходится использовать по нескольку раз на сайте. Можно даже передавать переменные в миксины, чтобы сделать

их более гибкими. Так же хорошо использовать миксины для вендорных префиксов. Пример для transform:

SCSS

```
@mixin transform($property) {
  -webkit-transform: $property;
  -ms-transform: $property;
  transform: $property;
}
.box { @include transform(rotate(30deg)); }
```

CSS

```
.box {
  -webkit-transform: rotate(30deg);
  -ms-transform: rotate(30deg);
  transform: rotate(30deg);
}
```

Чтобы создать миксин, мы здесь используем директиву @mixin и присваиваем ей имя. Мы назвали наш миксин transform. Кроме того, мы также используем переменную $property внутри круглых скобок, чтобы можно было передать любое преобразование, которое мы хотим. После создания миксина мы можете использовать его как объявление CSS, начинающееся с @include, за которым следует имя миксина.

**Расширение / Наследование**

Это одна из самых полезных функций SCSS. Используя директиву @extend, можно наследовать наборы свойств CSS от одного селектора другому. Это позволяет держать файл SCSS в «чистоте». На приведенном далее примере видно, как можно сделать стили оповещений об ошибках, предупреждениях и удачных исходах, используя другие возможности SCSS, которые идут рука-об-руку с расширением, классами-шаблонами. Класс-шаблон - особый тип классов, который выводится только при использовании расширения - это позволит сохранить ваш скомпилированный CSS чистым и аккуратным.

SCSS

```
/* This CSS will print because %message-shared is extended. */
%message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
```

```
}

// This CSS won't print because %equal-heights is never extended.
%equal-heights {
  display: flex;
  flex-wrap: wrap;
}

.message {
  @extend %message-shared;
}

.success {
  @extend %message-shared;
  border-color: green;
}

.error {
  @extend %message-shared;
  border-color: red;
}

.warning {
  @extend %message-shared;
  border-color: yellow;
}
```

CSS

```
 .message, .success, .error, .warning {
    border: 1px solid #ccc;
    padding: 10px;
    color: #333;
}

.success {
    border-color: green;
}

.error {
    border-color: red;
}

.warning {
```

```
    border-color: yellow;
}
```

Вышеуказанный код сообщает классам .message, .success, .error и .warning вести себя как %message-shared. Это означает, что где бы не вызывался %message-shared, то и .message, .success, .error и .warning тоже будут вызваны. Магия происходит в сгенерированном CSS, где каждый из этих классов получает css-свойства, как и %message-shared. Это позволит избежать написания множества классов в HTML элементах, которые, без использования расширения, было бы сложно просматривать для поиска требуемого.

Можно расширить большинство простых CSS селекторов прибавлением к классам-шаблонам в SCSS, однако, использование шаблонов - простейший способ быть уверенным, что вы не расширяете класс везде, где он используется в ваших стилях, что могло бы привести к непреднамеренным наборам стилей в вашем CSS.

Когда вы генерируете ваш CSS, то он будет выглядеть как пример, приведенный выше. Здесь важно обратить внимание на то, что %equal-heights не попадает в CSS, так как ни разу не был использован.

**Математические операторы**

Использовать математику в CSS очень полезно. Sass имеет несколько стандартных математических операторов, таких как +, -, *, / и %. Ниже приведен пример, в котором мы совершаем простые математические вычисления для расчета ширины aside и article.

SCSS

```
.container {
  width: 100%;
}

article[role="main"] {
  float: left;
  width: 600px / 960px * 100%;
}

aside[role="complementary"] {
  float: right;
  width: 300px / 960px * 100%;
}
```

Мы создали простую адаптивную модульную сетку, с шириной в 960 пикселей. Используя математические операторы, мы использовали полученные данные с пиксельными значениями и конвертировали их в процентные, причем без особых усилий. Скомпилированный CSS выглядит так:

CSS

```
.container {
  width: 100%;
}

article[role="main"] {
  float: left;
  width: 62.5%;
}

aside[role="complementary"] {
  float: right;
  width: 31.25%;
}
```

Подводя итоги, можно сказать, что препроцессор SASS помогает сделать код CSS проще и понятней, облегчает его восприятие, расширяет его функциональность, делает его написание ещё больше похожим на программирование благодаря применению констант, встроенных функций, переменных, смешанных стилей, наследования и так далее. Более того, препроцессор исключает повторяющиеся фрагменты кода. Главное, он облегчает работу и прщврояеи экономить время разработчику.

Может возникнуть вопрос, почему в заголовке данного раздела мы указали название препроцессора как SASS, а в примерах использовали код, озаглавливая его SCSS. Дело в том, что препроцессор SASS имеет два синтаксиса — SASS, использующий подход, принятый в языке Python, а именно — использование отступов для разделения элементов кода, и синтаксис SCSS, использующий фигурные скобки для этих же целей. В описании этой главы, как и для написания данного проекта, мною был выбран синтаксис SCSS.

### 2.3.4. Применение БЭМ.

Выбор использования препроцессора SCSS логично привёл меня к выбору методологии БЭМ, по крайней мере её нейминга, так как это органично сочетается с предоставленной препроцессором возможностью применить принцип вложенности.

Почему это важно?

БЭМ (Блок-Элемент-Модификатор) - это методология разработки веб-интерфейсов, предложенная Яндексом в 2013 году. Она позволяет создавать модульные и масштабируемые системы, которые легко поддерживать и развивать. Её использование позволяет создавать системы, содержащие модули, которые легко масштабируются и переиспользуются.

Суть идеи БЭМ заключается в том, что любой интерфейс разделяется на отдельные блоки. Блок - это основной кирпичик в при применении этой методологии. Он является независимым и самодостаточным компонентом. Блок можно использовать многократно в различных контекстах. При разработке, блок независим от других блоков, что даёт возможность разрабатывать и тестировать его отдельно, вплоть до использования различных команд, компаний и оборудования.

Блок может состоять из элементов и модификаторов. Элементы - это составные части блока, но, в отличие от блока, создаются для данного блока и не могут быть использованы вне его. Модификаторы, это некие режимы работы блока, которые легко переключаются, изменяя свойства или поведение всего блока. Это переключение не требует внесения изменений в код блока.

Присущий БЭМ механизм вложенности, позволяет организовать иерархию блоков, в структура и конфигурация которой может быть сложной, но легко управляемой. Используя модификаторы и стили, можно сравнительно просто управлять функциональностью блока.

Одним из главных преимуществ БЭМ является возможность повторного использования блоков. Блоки могут быть использованы в разных контекстах и на разных страницах без необходимости переписывать код. Это позволяет сократить время разработки и облегчить поддержку проекта.

Ещё, следует сказать о возможности повторного использования таких блоков. В силу присущей гибкости блоков, их можно использовать многократно на различных страницах и даже сайтах, просто изменяя их модификаторы, не изменяя сам код блока. Для получения такого эффекта, следует придерживаться следующих рекомендаций:

- Блок не должен влиять на свое окружение, т. е. блоку не следует задавать внешнюю геометрию (в виде отступов, границ, влияющих на размеры) и позиционирование.
- В CSS по БЭМ не рекомендуется использовать селекторы по тегам или id, а следует использовать классы.

Это позволит обеспечить независимость, при которой возможно повторное использование или перенос блоков с места на место.

БЭМ построен на следующих «трёх китах», суть — основных принципах:

1. **Блок** — независимый и логический компонент веб-страницы. Блок не зависит от внешнего контекста и может быть использован в других частях проекта. Блок имеет уникальное имя, которое описывает его функциональность.
2. **Элемент** — составная часть блока, которая не может использоваться независимо. Элементы находятся внутри блоков и отображаются в контексте блока. Имена элементов состоят из имени блока и имени элемента, разделённых двумя подчёркиваниями.
3. **Модификатор** — свойство блока или элемента, которое меняет его внешний вид или поведение. Модификаторы могут быть либо булевыми, либо ключ-значение. Булевые модификаторы указывают присутствие или отсутствие модификатора, а ключ-значение модификаторы указывают конкретные параметры модификации. Модификаторы добавляются к имени блока или элемента через

В целом, БЭМ - это мощный инструмент для разработки веб-интерфейсов, который позволяет создавать модульные и масштабируемые системы. Он упрощает разработку, улучшает поддержку и повышает переиспользуемость компонентов. БЭМ активно применяется в различных проектах Яндекса и других компаний, и оказывает существенное влияние на разработку веб-интерфейсов.

### 2.3.5. Браузерный инструмент разработчика.

Чем мне нравится разработка сайтов, так это тем, что можно сразу посмотреть результат, оценить его и изменить в нужном месте код, если потребуется. И здесь стоит сказать об инструменте разработчика в браузере. При создании сайта я использовал Chrome DevTools.

Сфера применения Google Dev Tools - это разработка и отладка веб-приложений. Она позволяет разработчикам искать и исправлять ошибки в своем коде, проводить профилирование производительности, анализировать сетевую активность и многое другое.

Основные составные компоненты Google Dev Tools включают:

1. Панель инструментов: это основной интерфейс, который предоставляет доступ ко всем функциям и инструментам Dev Tools. В нем можно выбрать различные панели, такие как "Elements", "Console", "Sources" и другие, для работы с разными аспектами приложения.

2. Панель Elements: эта панель позволяет визуально просматривать и редактировать DOM-дерево страницы. Она позволяет исследовать, изменять и отображать стили, атрибуты и содержимое элементов в реальном времени.
3. Панель Console: эта панель предоставляет консоль JavaScript, где можно выполнять JavaScript-код и отображать результаты, а также выводить сообщения об ошибках и предупреждения.
4. Панель Sources: эта панель позволяет отлаживать JavaScript-код в режиме реального времени. С помощью нее можно установить точки останова, выполнять шаги выполнения, наблюдать значения переменных и многое другое.
5. Панель Network: эта панель предоставляет информацию о сетевой активности при загрузке страницы. Она позволяет анализировать запросы и ответы, отслеживать время загрузки ресурсов и искать узкие места в производительности.

Некоторые особенности Google Dev Tools включают:

1. Инструменты для профилирования производительности: Dev Tools позволяет измерять и анализировать производительность страницы, идентифицировать проблемные места и улучшать скорость загрузки и отзывчивость приложения.
2. Отладка и исправление ошибок: с помощью Dev Tools разработчики могут отлаживать JavaScript-код, находить и исправлять ошибки, наблюдать значения переменных и объектов в реальном времени.
3. Манипулирование DOM и стилями: Dev Tools позволяет разработчикам динамически изменять DOM-структуру и стили элементов веб-страницы, чтобы проверять их внешний вид и поведение.
4. Эмуляция мобильных устройств: Dev Tools предоставляет возможность эмулировать различные мобильные устройства, чтобы проверять адаптивность и отзывчивость веб-приложений на разных разрешениях экрана и устройствах.

Эти особенности делают Google Dev Tools мощным инструментом для разработки и отладки веб-приложений.

### 2.3.6. Язык программирования Javascript.

Язык программирования JavaScript («JS») — это объектно-ориентированный язык сценариев, объекты которого встроены в программное обеспечение веб-браузера, например, Google Chrome, Microsoft Edge, Firefox, Opera и Safari. Для обеспечения функциональности это позволяет при загрузке страницы в браузере интерпретировать содержащиеся на веб-странице сценарии. В целях безопасности язык JavaScript не может считывать или

записывать файлы, за исключением файлов cookie, в которых хранится минимальный объем данных.

Самая первая реализация JavaScript была создана Бренданом Эйхом (Айком) (Brendan Eich) в компании Netscape. Этот язык программирования был впервые представлен в декабре 1995 года и первоначально назывался «LiveScript». Однако вскоре из коммерческих соображений LiveScript был переименован в JavaScript, получив соответствующую лицензию у Sun Microsystem.

До введения JavaScript браузер вызывал сценарии на стороне сервера, а в связи с долгим ответом снижалась производительность. Эта проблема решилась с помощью вызова сценариев на стороне клиента. Популярность языка JavaScript быстро росла. Но между компаниями Netscape и Microsoft возникли разногласия по поводу его лицензирования, поэтому Microsoft представила собственную версию языка под названием «JScript». Несмотря на то что новая версия JScript очень похожа на JavaScript, она имеет некоторые расширенные функции. В июне 1997 года Ecma International предложила стандартизировать JavaScript, и в результате появился «ECMAScript». Это помогло стабилизировать основные функции. Однако название не прижилось среди юзеров, поэтому язык программирования все же получил название JavaScript.

В отличие от других языков, которые можно использовать в веб-браузере, JavaScript не нужно загружать и устанавливать. В браузерах, поддерживающих JavaScript, он встроен в браузер и включен по умолчанию (это означает, что обычно вам нужно изменить настройки браузера, только если вы не хотите, чтобы браузер запускал JavaScript). Единственным исключением является то, что Internet Explorer также поддерживает vbScript таким же образом, и управление этими двумя языками осуществляется с помощью настроек, называемых «активными сценариями», а не настроек, конкретно ссылающихся на JavaScript.

Изначально JavaScript был создан, чтобы «оживить веб-страницы».
Программы на этом языке называются скриптами. Их можно встроить в HTML и запускать автоматически при загрузке веб-страницы.
Скрипты распространяются и выполняются как обычный текст. Для их запуска не требуется специальной подготовки или компиляции.
Это отличает JavaScript от другого языка — Java.

Сегодня JavaScript может выполняться не только в браузере, но и на сервере или на любом другом устройстве, которое имеет специальную программу,

называющуюся «движком» JavaScript. У браузера есть собственный движок, который иногда называют «виртуальная машина JavaScript».

Разные движки имеют разные «кодовые имена». Например: V8 – в Chrome, Opera и Edge, SpiderMonkey – в Firefox. Ещё есть «Chakra» для IE, «JavaScriptCore», «Nitro» и «SquirrelFish» для Safari и т.д. Эти названия полезно знать, так как они часто используются в статьях для разработчиков. Мы тоже будем их использовать. Например, если «функциональность X поддерживается V8», тогда «X», скорее всего, работает в Chrome, Opera и Edge.

Современный JavaScript – это «безопасный» язык программирования. Он не предоставляет низкоуровневый доступ к памяти или процессору, потому что изначально был создан для браузеров, не требующих этого.

Возможности JavaScript сильно зависят от окружения, в котором он работает. Например, Node.JS поддерживает функции чтения/записи произвольных файлов, выполнения сетевых запросов и т.д.

В браузере для JavaScript доступно всё, что связано с манипулированием веб-страницами, взаимодействием с пользователем и веб-сервером.

Например, в браузере JavaScript может:

- Добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили.
- Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.
- Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы (технологии AJAX и COMET).
- Получать и устанавливать куки, задавать вопросы посетителю, показывать сообщения.
- Запоминать данные на стороне клиента («local storage»).

Возможности JavaScript в браузере ограничены ради безопасности пользователя. Цель заключается в предотвращении доступа недобросовестной веб-страницы к личной информации или нанесения ущерба данным пользователя.

Примеры таких ограничений включают в себя:

- JavaScript на веб-странице не может читать/записывать произвольные файлы на жёстком диске, копировать их или запускать программы. Он не имеет прямого доступа к системным функциям ОС.

- Современные браузеры позволяют ему работать с файлами, но с ограниченным доступом, и предоставляют его, только если пользователь выполняет определённые действия, такие как «перетаскивание» файла в окно браузера или его выбор с помощью тега <input>.
- Существуют способы взаимодействия с камерой/микрофоном и другими устройствами, но они требуют явного разрешения пользователя. Таким образом, страница с поддержкой JavaScript не может незаметно включить веб-камеру, наблюдать за происходящим и отправлять информацию в ФСБ.
- Различные окна/вкладки не знают друг о друге. Иногда одно окно, используя JavaScript, открывает другое окно. Но даже в этом случае JavaScript с одной страницы не имеет доступа к другой, если они пришли с разных сайтов (с другого домена, протокола или порта).
- Это называется «Политика одинакового источника» (Same Origin Policy). Чтобы обойти это ограничение, обе страницы должны согласиться с этим и содержать JavaScript-код, который специальным образом обменивается данными.
- Это ограничение необходимо, опять же, для безопасности пользователя. Страница https://anysite.com, которую открыл пользователь, не должна иметь доступ к другой вкладке браузера с URL https://gmail.com и воровать информацию оттуда.
- JavaScript может легко взаимодействовать с сервером, с которого пришла текущая страница. Но его способность получать данные с других сайтов/доменов ограничена. Хотя это возможно в принципе, для чего требуется явное согласие (выраженное в заголовках HTTP) с удалённой стороной. Опять же, это ограничение безопасности.
- Подобные ограничения не действуют, если JavaScript используется вне браузера, например — на сервере. Современные браузеры предоставляют плагины/ расширения, с помощью которых можно запрашивать дополнительные разрешения.

### 2.3.7. Подключение шрифтов.

Существует несколько способов подключения шрифтов к странице сайта:

1. Локальное подключение шрифтов:

- Скачайте необходимый шрифт и сохраните его на сервере вашего сайта.
- Создайте CSS-файл, в котором будет определен путь к скачанному шрифту, например:

```
@font-face {
    font-family: 'Название_шрифта';
    src: url('путь_к_шрифту/шрифт.ttf') format('truetype');
}
```

- Подключите этот CSS-файл к вашей странице, добавив его в секцию <head>:

```
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Document</title>
        <link rel="stylesheet" type="text/css" href="путь_к_CSS-файлу/шрифты.css">
    </head>
```

- Теперь вы можете использовать этот шрифт в своих стилях:

```
body {
    font-family: 'Название_шрифта', sans-serif;
}
```

2. Использование Google Fonts:

- Откройте сайт Google Fonts (https://fonts.google.com/).
- Выберите нужный вам шрифт и нажмите кнопку "Select this font".
- Настройте параметры шрифта (жирность, начертание, размер).
- После настройки нажмите на иконку "Embed" и скопируйте предоставленный код.
- Вставьте скопированный код в вашу страницу внутри секции <head>:

```
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Document</title>
        <link rel="stylesheet" href="https://fonts.googleapis.com/css2?
        family=Название_шрифта&display=swap">
    </head>
```

- Теперь вы можете использовать этот шрифт в своих стилях:

```
body {
    font-family: 'Название_шрифта', sans-serif;
}
```

3. Подключение сторонних сервисов:

- Существуют различные сервисы, такие как Adobe Fonts или Typekit, которые предлагают шрифты и инструкции по их подключению.
- Зарегистрируйтесь на выбранном сервисе и получите доступ к необходимым шрифтам.
- Следуйте инструкциям по подключению шрифтов, предоставленным данным сервисом, чтобы использовать их на своей странице.

В данном проекте подходит первый способ подключения — локальное подключение, который и был реализован:

```
@font-face {
    font-family: 'Gilroy';
    src: url('Gilroy-Regular.eot');
    src: local('Gilroy Regular'), local('Gilroy-Regular'),
        url('Gilroy-Regular.eot?#iefix') format('embedded-opentype'),
        url('Gilroy-Regular.woff') format('woff'),
        url('Gilroy-Regular.ttf') format('truetype');
    font-weight: normal;
    font-style: normal;
}
```

**2.4. Анализ макета и выбор методов выполнения задачи.**

Основными моментами анализа макета сайта для планирования вёрстки являются определение структуры и компонентов, типографика, цветовая схема, графические элементы, медиа-элементы, адаптивность, взаимодействие и эффекты. В результате анализа происходит планирование структуры кода, определение возможных сложностей и взаимодействие с дизайнером для уточнений.

Итак, движемся по порядку в соответствии с намеченным планом.

1. Изучение общей структуры макета: анализ основных блоков, колонок, шапки и подвала, навигационных элементов и других ключевых компонентов.

- Поскольку в команде отсутствует дизайнер, то для выполнения поставленной задачи выбираем на просторах интернета макет. Подходящим источником макетов оказался телеграмм-канал «Макеты для вёрстки», расположенный по адресу [16]. Макет выполнен в программе Figma, находится по адресу [17].

- Макет имеет шапку и подвал, четыре информационных секции, одну секцию с видео, одну секцию с таблицей цен. Имеются кнопки «Начать обучение» и «Пройти тестирование», которые запускают прохождение тестирования на достаточную подготовленность для обучения в школе.

2. Оценка типографики: определение используемых шрифтов, их размеров, стилей и расположения. Также стоит заметить использование заголовков разных уровней, абзацев и списков.

   - Предложенные дизайнером шрифты относятся к категории рублёные шрифты, семейству Gilroy. В макете используется шесть начертаний шрифтов: Gilroy Regular (наиболее задействованное начертание), Gilroy SemiBold, Gilroy Medium, Gilroy Bold, Gilroy Heavy, Gilroy SemiBoldItalic. Шрифты получены на сайте [18], скачаны и установлены с использованием @font-face в файле style.scss.

3. Исследование цветовой схемы: анализ используемых цветов и их сочетаний, определение основного и дополнительных цветов, проверка доступности и контрастности.

   - В этом вопросе целиком доверимся дизайнеру, так как он требует некоторого количества опыта, а в данном случае мы говорим о первом опыте. Однако, здесь можно сделать вывод, что при выполнении заказов необходима плотная работа с дизайнером.

4. Анализ графических элементов: оценка использования изображений, иконок и других графических элементов, их размеров и размещения.

   - Анализ показал, что графические элементы макета доступны для экспорта, занимают немного места, так как большинство из них выполнены в графическом формате svg. Его основные преимущества: SVG (Scalable Vector Graphics) - это формат графики, который используется для отображения векторных изображений в веб-браузерах. Вот основные преимущества использования SVG:

     - Масштабируемость: SVG-изображения являются векторными, что означает, что они могут быть без потери качества масштабированы без ограничений. Это отличается от растровых изображений, которые могут потерять четкость и

разрешение при масштабировании.

- Оптимизированный размер файла: SVG-изображения имеют малый размер файла по сравнению с растровыми форматами, такими как JPG или PNG. Это может способствовать более быстрой загрузке страницы и экономии места на сервере.

- Доступность: SVG является текстовым форматом, что делает его доступным для редактирования и чтения непосредственно в коде. Это позволяет внести изменения в изображение без необходимости в редактировании графического файла.

- Анимация: SVG поддерживает анимацию и взаимодействие с помощью CSS и JavaScript. Это позволяет создавать интерактивные и динамические эффекты, такие как анимированные графики и реагирующие на действия пользователя.

- Поддержка различных устройств: SVG-изображения могут просматриваться на различных устройствах и экранах с разными разрешениями, и они все равно будут выглядеть одинаково хорошо и детализированно.

- Поддержка поисковых систем: SVG может содержать текст и метаданные, которые могут быть проиндексированы поисковыми системами. Это полезно для SEO (поисковая оптимизация) и позволяет лучше определить содержимое изображения.

- Возможность динамического редактирования: SVG может быть изменен и редактирован непосредственно в коде, без необходимости в редактировании исходного файла. Это облегчает и ускоряет процесс внесения изменений и создания новых версий изображений.

- Векторная графика: SVG представляет изображения как математические объекты, включающие в себя линии, кривые, формы и цвета. Это позволяет точно определить форму и расположение элементов, что полезно при создании сложных и детализированных изображений.

- В целом, SVG является мощным и гибким форматом для создания и отображения графики в интернете, обладая преимуществами масштабируемости, оптимизации

размера файла, доступности для редактирования и анимации, поддержки различных устройств и поисковых систем, а также возможности динамического редактирования и использования векторных объектов.

5. Определение медиа-элементов: проверка наличия видео, аудио, анимаций или других медиа-элементов в макете и их готовность для включения в верстку.

   - В макете присутствует секция, подразумевающая видео, но, поскольку само видео не представлено, то, при создании данной секции используем картинку — заглушку, выполним её центрирование в рамках фонового изображения телевизора, а затем, закончив вёрстку смежных секций, вставим для демонстрации работы сайта бесплатное видео с YouTube:

```
87          <div class="how-to__screen">
88              <iframe width="560" height="315" src="https://www.youtube.com/embed/
                idDE-QZyw6w?si=yVDj0hHwUpSJox2w"
89                  title="YouTube video player" frameborder="0"
90                  allow="accelerometer; autoplay; clipboard-write; encrypted-media;
                    gyroscope; picture-in-picture; web-share"
91                  allowfullscreen></iframe>
92              <!-- <img src="./img/how-to/howToScreen.png" alt="howToScreen"> -->
93          </div>
```

6. Изучение макета адаптивности: проверка наличия адаптивного дизайна и просмотр различных разрешений экранов для идентификации ключевых точек разрыва и поведения элементов.

   - В макете дизайнер предусмотрел использование данного сайта на мобильных устройствах. Более того, исходя из полученной информации при обучении, а также из множества источников в интернете, в последнее время предпочтение стало отдаваться технологии mobile first. Заключается она в следующем:

   - Mobile First - это подход к веб-разработке, который предполагает, что веб-сайт или приложение следует сначала разрабатывать для мобильных устройств, а затем масштабировать его для более крупных экранов. Основные принципы вёрстки по технологии Mobile First включают:

- Приоритет мобильных устройств: Mobile First подразумевает, что мобильные устройства имеют более высокий приоритет и они должны быть основным фокусом при разработке веб-страницы или приложения.

- Адаптивный дизайн: Веб-страницы должны быть разработаны с использованием адаптивного дизайна, чтобы автоматически адаптироваться к разным экранам и устройствам. Это достигается путем использования медиа-запросов CSS и гибкой сетки.

- Меньшие изображения: Поскольку скорость загрузки на мобильных устройствах является критическим фактором, изображения должны быть оптимизированы и использоваться в наиболее эффективном формате, чтобы снизить их размер и обеспечить быструю загрузку.

- Простота и минимализм: Интерфейс должен быть простым и минималистичным, чтобы упростить навигацию и обеспечить удобство использования на маленьких экранах.

- Удобный интерфейс с сенсорным управлением: Ассортимент устройств с сенсорным экраном изменил способ взаимодействия с веб-сайтами. Mobile First уделяет большое внимание гладкому и удобному взаимодействию с сенсорным управлением, таким как масштабирование и прокрутка.

- Оптимизация производительности: Mobile First обращает особое внимание на оптимизацию производительности, включая снижение количества запросов к серверу, сокращение использования ресурсов и минимизацию загрузки данных.

Последование этих принципов поможет создать удобный и отзывчивый интерфейс, который будет хорошо работать на мобильных устройствах и затем масштабироваться для больших экранов. Но, в нашем случае, с учётом отсутствия опыта разработки, будем изначально делать сайт для десктопов. По восприятию, от него проще переходить, реализуя адаптив для промежуточных размеров экранов, к мобильной реализации. Что следует знать про адаптивную вёрстку:

- Адаптивная верстка представляет собой подход в веб-разработке, при котором веб-сайт или приложение создается таким образом, чтобы автоматически

подстраиваться под разные экраны и устройства, обеспечивая оптимальное отображение и удобство использования для каждого пользователя.

- Основные методы адаптивной верстки:

  ◆ Медиа-запросы: Это инструмент CSS, который позволяет настраивать стили в зависимости от разрешения экрана или других параметров устройства, таких как ширина окна браузера или ориентация устройства (портретная или альбомная). Медиа-запросы помогают задавать различные правила стилей для разных размеров экрана.

  ◆ Гибкая сетка: Это техника, которая позволяет задавать элементы на странице с использованием относительных единиц измерения (проценты) вместо фиксированных пикселей. Это позволяет элементам пропорционально изменяться в зависимости от размера экрана, что позволяет создать удобную и гибкую структуру страницы.

  ◆ Изображения с изменяемым размером: Вместо использования фиксированного размера изображений, адаптивная верстка позволяет задавать размер изображения относительно контейнера, в котором оно выводится. Это позволяет изображениям адаптироваться к разным размерам экрана, сохраняя при этом свое соотношение сторон и качество.

  ◆ Скрытие и перенос контента: Этот метод заключается в скрытии или переносе некоторых элементов контента, чтобы обеспечить лучшую читаемость и навигацию на более малых экранах. Например, некоторые элементы могут быть скрыты или заменены иконками, а меню навигации может быть перенесено в выпадающий список.

- Преимущества адаптивной верстки:

  ◆ Удобство использования для пользователей на разных устройствах: Веб-сайт или приложение, созданные с использованием адаптивной верстки, отлично работают на разных устройствах - от настольных компьютеров до мобильных устройств. Это обеспечивает удобство использования для пользователей и улучшает их общий опыт.

◆ Лучшая оптимизация для поисковых систем: Адаптивный веб-сайт имеет один URL и одну HTML-структуру, что упрощает его индексацию поисковыми системами. Это может улучшить его видимость в поисковых результатах и привести к большему количеству органического трафика.

◆ Экономия времени и ресурсов: Поскольку адаптивная верстка позволяет создать один сайт, который работает на всех устройствах, нет необходимости разрабатывать и поддерживать несколько версий сайта для разных платформ. Это упрощает процесс разработки и экономит время и ресурсы разработчика.

◆ Возможность легкой настройки и обновления: Адаптивный сайт или приложение проще настраивать и обновлять, поскольку изменения отражаются на всех устройствах одновременно. Это упрощает процесс поддержки сайта.

7. Оценка взаимодействия и эффектов: проверка наличия интерактивных элементов, таких как ссылки, кнопки, выпадающие списки и других элементов, а также дизайн состояний hover, active и disabled.

- Основные интерактивные элементы на сайте, это якорные ссылки, кнопки и модальные окна, которые также имеют кнопки и блоки, выполняющие роль кнопки («крестик» для закрытия окна).

8. Планирование структуры кода: на основе описанных выше аспектов макета, необходимо определить нужные HTML-теги, классы CSS, id и другие элементы для построения адекватной структуры кода.

- На этом этапе необходимо решить вопрос с целесообразностью применения того или иного фреймворка. На курсе мы изучали фреймворк Vue, и нам была дана достаточная информация и практический опыт на лекциях, семинарах и домашних заданиях. Поэтому, выбор фреймворка очевиден. Однако, необходимо рассмотреть целесообразность применения фреймворка вообще для конкретного нашего случая. Итак, что нам даст применение Vue в нашем случае:

◆ Vue.js - это прогрессивный JavaScript-фреймворк, используемый для разработки пользовательского интерфейса. Он предоставляет инструменты для создания отзывчивых и интерактивных веб-приложений.

◆ Одной из особенностей Vue.js является его простота в освоении. Он имеет маленький размер и понятный API, что позволяет разработчикам быстро начать работу с ним. Vue.js также поддерживает поэтапную интеграцию, что означает, что вы можете постепенно внедрять его в существующий проект без необходимости переписывать всю кодовую базу.

◆ Еще одной важной особенностью Vue.js является его реактивная система. Он использует виртуальный DOM (VDOM), который позволяет обновлять только необходимые компоненты, улучшая производительность приложения. Также Vue.js автоматически отслеживает и перерисовывает компоненты, когда изменяются данные, связанные с ними, что делает взаимодействие с данными более простым и интуитивным.

◆ Vue.js также обладает мощными возможностями компонентной разработки. Он позволяет создавать многоразовые компоненты, которые могут быть использованы в разных частях приложения. Компоненты Vue.js также могут быть легко объединены и наследованы, что способствует повторному использованию кода и упрощает его поддержку.

◆ Еще одной интересной особенностью Vue.js является его экосистема. Он имеет широкий спектр плагинов и расширений, которые помогают улучшить функциональность и производительность приложения. Кроме того, Vue.js имеет активное сообщество разработчиков, которые постоянно работают над улучшением и развитием фреймворка, а также создают множество учебных ресурсов и документации для новых пользователей.

◆ В целом, Vue.js предлагает простоту, гибкость и высокую производительность в процессе разработки веб-приложений. Благодаря своим мощным функциям и поддержке сообщества разработчиков, он становится все более популярным выбором для создания отзывчивых и интерактивных пользовательских интерфейсов.

- Несмотря на свою относительную новизну, Vue.js имеет некоторые преимущества по сравнению с другими фреймворками и использованием ванильного JavaScript:

◆ Легкость изучения и интеграции: Vue.js имеет простой и интуитивно понятный синтаксис, что упрощает его изучение. Он также легко интегрируется в

существующие проекты, что позволяет постепенно переходить к его использованию.

◆ Гибкость и масштабируемость: Vue.js предоставляет гибкую архитектуру, которая позволяет разработчикам создавать сложные страницы и веб-приложения. Он также позволяет создавать компоненты, которые можно переиспользовать в разных проектах, что делает его масштабируемым.

◆ Высокая производительность: Vue.js использует виртуальный DOM, который обеспечивает быструю отрисовку и обновление элементов на странице. Он также имеет механизмы для оптимизации производительности, такие как асинхронная компиляция шаблонов и ленивая загрузка компонентов.

◆ Разработчику дружелюбный экосистем: Vue.js имеет активное сообщество разработчиков и богатую документацию, которые облегчают процесс разработки. Также существует множество готовых компонентов и плагинов, которые можно использовать для быстрой разработки.

◆ Эффективное управление состоянием: Vue.js предлагает инструменты для эффективного управления состоянием приложения. Он имеет встроенную систему реактивности, которая автоматически обновляет компоненты при изменении состояния данных.

◆ Если же сравнивать с ванильным JavaScript, Vue.js предлагает более эффективное управление состоянием, модульность, переиспользуемость кода и более простой синтаксис для создания пользовательского интерфейса. Он также предоставляет более мощные инструменты для отладки и разработки, такие как расширения для браузера и среду разработки Vue Devtools. Однако, использование ванильного JavaScript может быть более простым для небольших проектов или тех, кто предпочитает не зависеть от сторонних фреймворков.

- Несмотря на множество преимуществ и популярность, у Vue.js также есть некоторые недостатки по сравнению с другими фреймворками и использованием ванильного JavaScript:

◆ Объемность: Vue.js включает в себя множество функций и возможностей, что делает его относительно объемным. Это может привести к небольшой задержке при

загрузке страницы, особенно если проект не оптимизирован.

◆ Относительная новизна: Vue.js является более новым фреймворком по сравнению с Angular или React, что может привести к меньшему количеству ресурсов, инструкций и плагинов, доступных для разработчиков. Это может стать проблемой, если вы столкнетесь с каким-либо уникальным или сложным запросом.

◆ Меньшее сообщество сообщества: поскольку Vue.js относительно новый, его сообщество разработчиков не так велико, как у других популярных фреймворков. Это может ограничить доступ к учебным материалам, форумам для помощи в решении проблем или готовым решениям.

◆ Сложность масштабирования: Vue.js предназначен для создания простых и средних по сложности приложений. Однако, если ваш проект становится сложным и развивается со временем, масштабирование может стать сложной задачей. Некоторые другие фреймворки, такие как Angular, предлагают более мощные инструменты для управления сложностью проекта.

◆ Потенциальные проблемы с обратной совместимостью: поскольку Vue.js все еще активно развивается и находится в стадии версионирования, могут возникнуть проблемы с обратной совместимостью между разными версиями фреймворка. Это может вызвать сложности при переходе между версиями или работе с проектами, созданными на разных версиях Vue.js.

◆ Ограниченная поддержка корпоративными компаниями: поскольку Vue.js является более новым фреймворком, некоторые крупные корпоративные компании могут предпочесть использовать более устоявшиеся и популярные фреймворки, такие как Angular или React. Это может снизить доступность вакансий или проектов, связанных с Vue.js.

В целом, несмотря на эти недостатки, Vue.js все равно является мощным и удобным фреймворком для разработки веб-приложений. Выбор между использованием Vue.js и другими фреймворками или ванильного JavaScript зависит от потребностей, опыта и предпочтений.

Рассмотрев аспекты применения Vue.js, приходим к выводу, что в нашем случае, с учётом

того, что сайт одностраничный, что не даёт преимуществ использования модульности Vue.js, рациональнее будет использовать ванильный Javascript.

9. Оценка возможных сложностей: на этом этапе следует оценить возможные проблемы в верстке, такие как нестандартные компоненты, сложности с медиа-элементами или сложной адаптивностью, и подготовить план действий для их решения.

   - Рассмотрев данный аспект подготовки к разработке, можно сделать вывод, что нестандартные компоненты дизайнером не предусмотрены, что облегчает задачу для начинающего веб-разработчика. Оценить сложность включения медиа-элементов (а это вставка видео на сайт), не предоставляется возможным, так как дизайнер не представил информации о том видео, которое предполагается использовать на этом сайте. В данной ситуации очень важно поддерживать тесный контакт с дизайнером и, вероятно, с заказчиком сайта для получения дополнительной информации.

10. Коммуникация с дизайнером: в процессе анализа макета могут возникнуть вопросы или требования для уточнений у дизайнера, поэтому важно поддерживать открытую коммуникацию и согласовывать детали.

    - Данный пункт перекликается с выводами, сделанными в предыдущем пункте. Некоторые детали макета требуют дополнительного согласования с дизайнером.

11. Подготовка файлов и ресурсов: на основе проведенного анализа, создаются необходимые файлы (HTML, CSS, изображения и т. д.) и ресурсы для дальнейшей вёрстки. Также, при необходимости, проводится оптимизация изображений для улучшения производительности сайта.

    - Достижение данного пункта означает, что мы приступаем непосредственно к вёрстке нашего сайта.

Основные этапы верстки сайта включают в себя следующие этапы:

1. Сбор информации и планирование: В первую очередь, необходимо провести исследование и определить цели и требования к сайту. Это включает в себя изучение целевой аудитории, конкурентов, а также определение структуры и функциональности сайта.

2. Создание макета сайта: Основная идея этого этапа - разработать концепцию внешнего вида и компоновки сайта. Здесь важно учесть фирменный стиль, цветовую схему, шрифты и композицию веб-страницы. Макеты могут быть созданы с помощью инструментов дизайна, таких как Adobe Photoshop или Sketch.

3. Написание HTML и CSS: Для создания верстки сайта необходимо написать код на языке гипертекстовой разметки (HTML) и каскадных таблиц стилей (CSS). HTML определяет структуру и содержимое страницы, а CSS устанавливает стиль и внешний вид страницы. Эти языки используются для определения элементов, их позиционирования и форматирования.

4. Разработка адаптивности: В современном интернете важно, чтобы сайт хорошо отображался на разных устройствах и экранах. Поэтому, важным этапом верстки является разработка адаптивного дизайна. Для этого используются медиа-запросы CSS, которые позволяют изменять стили в зависимости от разрешения экрана.

5. Добавление интерактивных элементов: На этом этапе добавляются интерактивность и динамика на сайт. Для этого могут использоваться JavaScript и его библиотеки. Это включает в себя создание слайдеров, выпадающих меню, форм обратной связи и других элементов, которые позволяют пользователям взаимодействовать с сайтом.

6. Тестирование и оптимизация: Важно проверить, правильно ли работает весь функционал сайта и нет ли ошибок. Также, следует убедиться, что сайт загружается быстро и имеет хорошую производительность. В этом помогает тестирование на различных браузерах и устройствах.

7. Публикация и поддержка: После тестирования и оптимизации сайт готов к публикации. Он может быть размещен на хостинге или опубликован на сервере. После публикации необходимо осуществлять регулярную поддержку и обновление сайта для обеспечения его безопасности и актуальности.

После выполнения вёрстки, проверки её соответствия макету, задумкам дизайнера и требованиям заказчика, следует приступать к работе с интерактивом. Но, надо иметь в виду, что эта работа должна вестись уже на этапе вёрстки, что заключается в создании граммотной и удобной для программирования структуры сайта, определения использования тех или иных , применения правильного нейминга.

Основные этапы и особенности создания интерактивности сайта включают:

1. Планирование и структурирование: На этом этапе определяются цель и задачи сайта, определяется аудитория и планируется его структура. Решается, какие функции и взаимодействие будут реализованы на сайте.

2. Дизайн и визуальное оформление: Создается дизайн сайта, который включает в себя визуальные элементы, цветовую схему, шрифты и макеты. Важно учесть, что интерактивные элементы должны быть понятны и привлекательны для пользователей.

3. Front-end разработка: На этом этапе разработчики используют HTML, CSS и JavaScript для создания интерактивности сайта. Они создают веб-страницы, настраивают анимации, формы ввода данных, выпадающие списки и другие интерактивные элементы.

4. Back-end разработка: Back-end разработчики создают серверную часть сайта, которая обрабатывает запросы от пользователей и взаимодействует с базой данных. На этом этапе реализуются функции, связанные с пользовательской аутентификацией, обработкой форм, отправкой электронных писем и другими бизнес-логиками.

5. Тестирование и отладка: Важный этап, на котором проводится тестирование сайта на различных браузерах и устройствах, чтобы убедиться, что все интерактивные элементы работают должным образом. Выявленные ошибки исправляются и проводятся оптимизации для улучшения производительности сайта.

6. Запуск и поддержка: После успешного тестирования сайт готов к запуску. Затем следует регулярная поддержка и обновление сайта, чтобы гарантировать его безопасность, актуальность и функциональность. Новые интерактивные функции могут быть добавлены или модифицированы, и процесс создания интерактивности сайта становится непрерывным.

Здесь следует сказать о предстоящей, в реальной ситуации, поддержке созданного сайта. Осуществление поддержки и сопровождения сайта включает в себя ряд действий, которые направлены на обеспечение его бесперебойной работы, актуальности контента и безопасности.

1. Техническая поддержка: поддержка сайта начинается с обеспечения его надежной

работы, включая проверку и обновление программного обеспечения, устранение возникших ошибок и сбоев, регулярное резервное копирование данных, мониторинг нагрузки на сервер и скорости загрузки страниц.

2. Обновление контента: регулярное обновление контента на сайте помогает привлечь и удерживать пользователей. Это включает добавление новых статей, новостей, продуктов, услуг и т.д. Кроме того, необходимо обновлять информацию о контактах, адресе, рабочем времени и других важных данных.

3. SEO-оптимизация: чтобы сайт был видимым для поисковых систем и привлекал больше органического трафика, требуется оптимизация страницы. Это включает в себя использование правильных ключевых слов, улучшение скорости загрузки страницы, создание уникальных мета-тегов, размещение ссылок на другие страницы сайта и внешних ресурсов.

4. Безопасность: обеспечение безопасности сайта является одним из важных заданий. Это включает регулярное обновление программного обеспечения, установку защитных механизмов, мониторинг активности пользователей, резервное копирование данных и защищенное хранение пользовательской информации.

5. Аналитика: для эффективного управления сайтом необходимо анализировать его производительность и использование ресурсов. Веб-аналитика предоставляет данные о количестве посетителей, их поведении на сайте, эффективности маркетинговых кампаний и другую полезную информацию. Это помогает принимать обоснованные решения для улучшения производительности сайта.

В целом, поддержка и сопровождение сайта являются непрерывным процессом, который включает множество задач. Цель состоит в том, чтобы обеспечить безоперационное функционирование сайта, его актуальность и высокий уровень безопасности.

# 3. Заключительная часть

## 3.1. Подведение итогов

Была изучена и сделана оценка целесообразности перехода на новую профессию с точки зрения её перспективности. Приведенные результаты исследований из сторонних источников однозначно указывает на перспективность выбранного направления и изучения специальностей в IT – сфере, в частности, веб-разработке.

Оценена значимость выбранной специальности для общества. Быстрое распространение веб-сервисов, различных медиасервисов, большая вовлечённость в эти процессы молодёжи не оставляет сомнений в возможности использовать эти сервисы на пользу обществу.

Был изучен макет сайта, проведен его анализ и намечены основные элементы макета и оптимальные способы их вёрстки.

Был рассмотрен минимально необходимый набор инструментов для выполнения разработки данного сайта.

Были рассмотрены и проанализированы основные методики, необходимые для получения результата. Сделаны выводы о целесообразности применения фреймворка Vue.js в данном конкретном случае.

Была выполнена вёрстка сайта по выбранному макету и создана минимально необходимая его интерактивность.

Были рассмотрены возможные сложности при реализации выбранного макета.

Сформулированы дополнительные рекомендации начинающему frontend – программисту для преодоления подобных трудностей в дальнейшем. Они изложены в разделе Заключение.

### 3.2. Заключение

На основании проведенной работы были определены некоторые моменты, которые можно порекомендовать начинающему frontend-разработчику. Не претендую на исчерпывающую инструкцию, тем более, что интернет такими рекомендациями переполнен, но исхожу из того, что личный опыт всегда имеет ценность, и лишь поделюсь теми моментами, которые следует учитывать при начале работы по созданию сайтов.

1. Наверное, главный совет по итогам данной работы — поддерживать плотный контакт с дизайнером и, если необходимо, с заказчиком.
2. При дальнейшем изучении и углублении в специальность, больше уделять практическим примерам, рассматриваемым на лекции. Обязательно прорабатывать их при повторном просмотре записи лекции, набирая демонстрируемый код и заставляя его работать.
3. Отдача от семинара значительно больше, если совет, предложенный в предыдущем пункте, будет использован до семинара, что позволит более эффективно закреплять на семинаре практические навыки.
4. Очень много информации можно и нужно получать в интернете, делая упор на техническую документацию, предлагаемую разработчиками используемых инструментов и методологий.
5. Активно изучать библиотеки, использовать API и прочие сервисы и наработки более опытных программистов.

## Список литературы


1. Паспорт национальной программы «Цифровая экономика Российской Федерации». URL: http://government.ru/info/35568/ (дата обращения: 06.05.2021).
2. Отрасль информационных технологий: некоторые важные факты за 6 лет. URL: http://government.ru/info/32158/ (дата обращения: 06.05.2021).
3. Новости цифровой трансформации, телекоммуникаций, вещания и ИТ. URL: https://www.comnews.ru/content/118771/2019-03-29/oleg-podolskiy-upravlyayushchiy-direktor-centra-kompetenciy-po-kadram-cifrovoy-ekonomiki-nacproekta-cifrovaya-ekonomika-rossiyskoy (дата обращения: 06.05.2021).
4. Yandex.Practicum and Analytical service HeadHunter (2019). IT: overview of the job market. URL: https://yandex.ru/company/researches/2019/it-jobs (дата обращения: 06.05.2021).
5. Сайт российской компании интернет-рекрутмента. URL: https://hh.ru/ (дата обращения: 06.05.2021).
6. Сайт SuperJob (IT-сервис по поиску работы и подбору сотрудников). URL: https://russia.superjob.ru/ (дата обращения: 06.05.2021).
7. Information Technology Curricula 2017. Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology. URL: https://www.acm.org/binaries/content/assets/education/curricula-recommendations/it2017.pdf (дата обращения: 06.05.2021).
8. Портал Федеральных государственных образовательных стандартов высшего образования. URL: http://www.fgosvo.ru/ (дата обращения: 06.05.2021).
9. Кузнецова Л. Г., Худжина М. В. Проблемы разработки системы оценки деятельности преподавателя вуза в условиях реализации образовательных стандартов // Интернет-журнал «Науковедение». 2013. № 1 (14). URL: https://naukovedenie.ru/PDF/19pvn113.pdf (дата обращения: 06.05.2021).
10. Каракозов С. Д., Петров Д. А., Худжина М. В. Проектирование основных образовательных программ в условиях приведения действующих ФГОС высшего образования в соответствие с профессиональными стандартами // Преподаватель XXI век. 2015. № 2–1. С. 9–23.
11. Маршалова Е. С., Рыжова Н. И. Тенденции профессиональной подготовки специалиста в области корпоративных маркетинговых коммуникаций с учетом конъюнктуры рынка труда и взаимодействия с работодателями // Преподаватель XXI век. 2019. № 2–1. С. 34–46.

12. Приказ Министерства труда России от 18 января 2017 года № 44н «Об утверждении профессионального стандарта «Разработчик Web и мультимедийных приложений». URL: http://www.consultant.ru/document/cons_doc_LAW_212176/ (дата обращения: 06.05.2021).
13. Sukhomlin V., Zubareva E. Standardization of IT education based on curriculums at the present stage // CEUR Workshop Proceedings. Selected Papers of the 11th International Scientific-Practical Conference Modern Information Technologies and IT-Education, SITITO 2016. 2016. P. 40–46.
14. Training of IT-specialists in russian and european higher education: a comparative study / S. D. Karakozov, M. V. Khudzhina, S. I. Gorlov et al. // icCSBs 2019 – The Annual International Conference on Cognitive – Social, and Behavioural Sciences. Conference processing. Edited by S. V. Ivanova, I. M. Elkina. 2019. P. 181–190.
15. Сайт Учебного Центра «Специалист» при МГТУ им. Н. Э. Баумана. URL: https://www.specialist.ru/ (дата обращения: 06.05.2021).
16. Телеграмм-канал «Макеты для вёрстки». URL: https://t.me/figma2html.
17. Макет сайта. URL: https://www.figma.com/file/wO7oUh1UZ8CTasIfO10MOr/YtYt?node-id=0%3A1
18. Сайт шрифтов Web Fonts Pro. URL: https://webfonts.pro/base-web-fonts/sans-serif-grotesque/369-gilroy.html
19. Официальный сайт фреймворка Vue.js: URL: https://v3.ru.vuejs.org/

## Приложения

1. Проект расположен на GitHub по адресу .
   URL: https://github.com/SemyIgor/DIP_PROJECT.git
2. Результат на GitHub Pages. URL: https://semyigor.github.io/DIP_PROJECT/