Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОЛОГИЙ УПРАВЛЕНИЯ ОБЪЕКТНОЙ МОДЕЛЬЮ ВЕБ-ДОКУМЕНТА

АвторШвалева Дарья Сергеевна _____

Направление подготовки (специальность) 09.04.02
Информационные системы и технологии

Квалификация магистр

Руководитель ВКРГосударев И.Б., к.педагог.н., доцент _____

К защите допустить

Руководитель ОП_____ _____

«_____»_____ 20 ____г.

Санкт-Петербург, 2019 г.

Студент Швалева Дарья СергеевнаГруппа Р4262Факультет ПИиКТ

Направленность (профиль), специализация 09.04.02 Информационные системы и технологии, Веб-технологии

Консультант (ы):
а) Государев И.Б., к.педагог.н., доцент _____ _____

ВКР принята «____»_____20 ____г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой _____

Дата защиты « 3 »июня 2019г.

Секретарь ГЭК _____ _____

Листов хранения 63

Демонстрационных материалов/Чертежей хранения _____

Оглавление

Введение

Развитие технологий и общества неизменно связано с большим объемом информации, что приводит к необходимости контроля и управления ею. Для грамотного управления информационными процессами необходим набор готовых классов, процедур, функций, структур и констант в программных средах. Развитие веб-платформ и веб-технологий в целом влечет за собой активное развитие программных интерфейсов приложений и интерфейсов прикладного программирования. Вместе с этим возрастает веб-нагрузка, что приводит к повышению требований к производительности сайтов и приложений.

Развитие веб-технологий и пользовательских интерфейсов актуализировало проблему получения доступа к содержимому HTML, XHTML и XML-документов, а также изменения содержимого, структуры и оформления таких документов.

Актуальность темы определяется тем, что все файлы для веб-страницы, такие как HTML, XML и CSS перенаправляются в веб-браузер, где они компилируются из текстовой модели в объектную. Это может быть актуальным и для всех видов кода. Браузер работает с объектной моделью, а не с самими файлами. Браузер взаимодействует с моделью, которая называется Document Object Model. Эта модель объединяет объекты в документах. В DOMведены все необходимые значения и стили для требуемого HTML и CSS-кода. Определенная модель для конкретной веб-страницы именуется DOM-деревом страницы.

В настоящее время не все веб-браузеры должны работать с DOM, чтобы компилировать HTML-документ. Однако, для скриптов, написанных на JavaScript, которым необходимо следить или изменять веб-страницу динамически, необходима Document Object Model.Следовательно, можно сказать, что DOM – это инструмент, благодаря которого JavaScript наблюдает за содержимым HTML-страницы и состоянием браузера. Зачастую JavaScript используется в виде встраиваемого язык для обеспечения доступа к объектам приложений.

Актуальность темы подтверждается многочисленными научными исследованиями, в том числе и [9; 10; 11]. Например, Ю.С Медведев в своей

работе говорит о том, что «В настоящее время веб-разработчики могут выбирать из широкого круга библиотек и фреймворков JavaScript, обеспечивающих возможность манипуляции DOM, но сложно остановиться на одной конкретной.» [10]. Это связано с большим количеством различных методологий управления объектной моделью веб-документа, которые решают большинство задач разработки и имеют свои преимущества и недостатки, однако изучению данного вопроса уделено достаточно мало внимания, как правило авторы останавливаются на одной выбранной технологии, описывая ее типовое применение, не проводя сравнительного анализа с другими методологиями в своем исследовании. В открытом доступе находится минимальное количество разбросанной по всей сети Интернет информации по данному вопросу, что обуславливает ценность данного исследования.

Цель ВКР: провести сравнительный анализ методологий управления объектной моделью веб-документа.

Задачи ВКР:

1.Проанализировать найденные источники по теме ВКР.
2.Изучить характеристики и возможности методологий управления объектной моделью веб-документа.
3.На основе изученного сделать сравнительные выводы, на основе которых выбрать методологии и критерии для практического анализа.
4.Написать три небольших ToDo-приложений, с помощью jQuery, React.js и JavaScript.
5.Сравнить полученные при практическом анализе показатели.
6.Оформить ВКР.

Объектом данной работы являются методологии управления объектной моделью веб-документа.

Предметом исследования является веб-разработка.

В процессе выполнения работы применялись следующие методы научного исследования: анализ, сравнение, обобщение.

Научная новизна данного исследования заключается в том, что были описаны характеристики и возможности методологий управления объектной

моделью веб-документа, которые могут быть использованы при выборе методологии для веб-разработки.

Теоретическая значимость данной работы состоит в том, что тщательным образом изучены найденные источники по теме ВКР, а также методологии управления объектной моделью веб-документа, такие как jQuery, React.js и JavaScript. На основе изученного выделены и сравнены общие характеристики и возможности методологий, которые помогли при дальнейшем практическом сравнении. На основе практического анализа получены результаты и сделаны выводы.

Практическая значимость работы заключается в том, что в результате, на основе разработанных ToDo-приложений произведен сравнительный анализ методологий управления объектной моделью веб-документа. В качестве результата сформулирована рекомендация по методологиям управления объектной моделью веб-документа с целью повышения производительности приложений, увеличения скорости и упрощения процесса разработки.

Положения, выносимые на защиту:

1. Причиной малой изученности методологий управления DOM является большое количество различных аналогов.
2. Выбор методологии может зависеть от требований к разрабатываемому продукту.
3. В ходе исследований выявлены критерии, сравнительный анализ которых позволил сформулировать рекомендации относительно использованных в работе методологий управления DOM.

Результаты, представленные в данной работе, обсуждались на следующих конференциях:

1. XLVII Научная и учебно-методическая конференция Университета ИТМО 30 января – 2 февраля 2018 года.
2. XLVIII Научная и учебно-методическая конференция Университета ИТМО 29 января – 1 февраля 2019 года.

По результатам данной работы были опубликованы следующие статьи:

1. Швалева Д.С. Сравнительный анализ методологий управления объектной моделью веб-документа. Этап 1. «Анализ зарубежных и отечественных источников по теме исследования» - Альманах научных работ молодых ученых Университета ИТМО – Университет ИТМО, 2018, Том 7, с.320
2. Швалева Д.С. Критерии сравнения методологий управления объектной моделью веб-документа// Альманах научных работ молодых ученых Университета ИТМО (принята в печать) — 2019.

ГЛАВА 1. АНАЛИЗ ЗАРУБЕЖНЫХ И ОТЕЧЕСТВЕННЫХ ИСТОЧНИКОВ

1.1Анализ общей информации о Document Object Model

Document Object Model – это интерфейс прикладного программирования для документов HTML и XML. Он представляет логическую структуру документов, способ доступа и управления документом. В спецификации DOM термин «документ» используется в широком смысле все чаще, XML применяется как способ описания разных видов информации, которые могут храниться в различных системах [16]. XML передает эти данные в виде документов, а DOM применяется для управления этими данными. Благодаря Document Object Model появилась возможность создавать документы, двигаться по их структуре, добавлять, удалять или изменять элементы и их содержимое. Все, что представляется в документе HTML или XML, может быть доступно с помощью Document Object Model.

Модель DOM не ограничивает структуру документа. С помощью данной модели каждый документ может быть изображен в виде структуры, которая напоминает структуру дерева узлов, в котором каждый узел изображает собой элемент или любой другой объект. В основном, DOM объединяет веб-страницу с языками программирования либо с языками, с помощью которых описывают сценарии.

DOM основан на объектной структуре, которая напоминает структуру документов, которые она формируют. Примере таблицы, взятой из документа HTML [17] приведен на рисунке 1.

```
<TABLE>
<TBODY>
  <TR>
    <TD>Один</TD>
    <TD>Два</TD>
  </TR>
  <TR>
    <TD>One</TD>
    <TD>Two</TD>
  </TR>
</TABLE>
</TBODY>
```

Рисунок 1– Представление DOM-таблицы

На рисунке 2 изображено представление DOM-таблицы следующим образом:

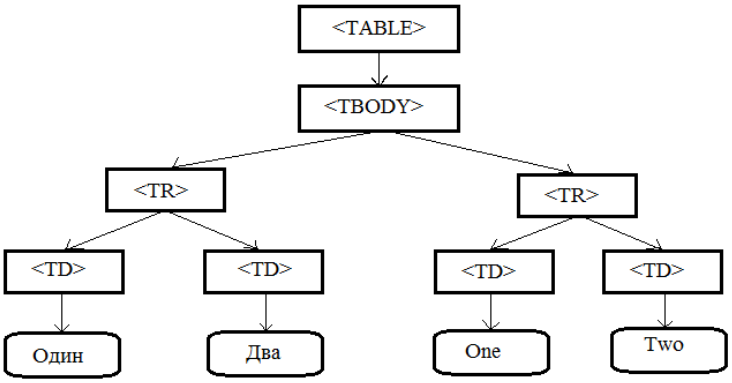

Рисунок 2– Представление DOM таблицы примера

Анализ исследования Консорциума Всемирной паутины [17] показал, что DOM документы обладают логической структурой, которая напоминает дерево, а точнее, как «рощу», в котором может содержаться больше одного дерева. Тем не менее, DOM не указывает, каким образом должны быть реализованы документы, а также не указывает, как будут выполняться отношения между объектами. DOM является логической моделью, которая может быть реализована любым удобным способом. В одной из спецификаций используется модель структуры терминов для описания древовидного представления документа. Также используется термин «дерево», когда речь идет о расположении этих информационных элементов, которые могут быть достигнуты с помощью методов «древовидной ходьбы». Одним из главных свойств структуры DOM является структурный изоморфизм: если при разработке представления одного документа используются

две реализации DOM, они будут формировать одну и ту же структурную модель в соответствии с XML.

Имя «объектная модель документа» было выбрано, так как оно является «объектной моделью» в традиционном объектно-ориентированном смысле проекта: документы создаются с использованием объектов, а модель содержит не только структуру документа, но захватывает поведение документа и объектов, из которых он построен. Значит, узлы на приведенной выше блок-схемы не изображают структуру данных, а изображают объекты, которые обладают функциями и идентичностью. Согласно исследованию [17], в виду объектной модели DOM определяет:

1.Интерфейсы и объекты, используемые для создания и управления документом.
2.Семантику интерфейсов и объектов, в том числе поведение и атрибуты.
3.Отношения и взаимодействия между этими интерфейсами и объектами.

В самом начале разные браузеры обладали собственными моделями документов, которые не обладали коммуникабельностью с другими. Для гарантии взаимной и обратной кроссбраузерности эксперты консорциума W3C систематизировали эту модель по уровням, на основе чего была разработана своя классификация для каждого отдельного уровня.

Консорциум Всемирной паутины (W3C) – компания, которая создает и вводит технологические стандарты для Всемирной паутины. Во главе Консорциума стоит сэр Тимоти Джон Бернерс-Ли, автор большого количества продуктов в области информационных технологий [26]. Все созданные стандарты соединены в общую группу, которая несет название «W3C DOM».

Госты W3C DOM и WHATWG DOM разрабатывают основы DOM, которые в свое время уже реализованы в современных браузерах. Большинство браузеров рекомендуют расширения за гранью данного стандарта, поэтому необходимо испытывать работоспособность возможностей DOM для каждого определенного браузера.

Анализ исследований [17; 18; 19] показал о наличии уровней DOM.

На данный момент устоявшимся уровнем спецификаций DOM является Уровень 2, но все неопределенные части спецификаций Уровня 3 являются рекомендуемыми W3C.

1.Уровень 0.

Состоит из определенных частей модели DOM, которые имели место быть появления Уровня 1. Стоит обратить внимание, что все модели официально не являются стандартами DOM, обнародованными W3C, а вероятнее являются информацией о том, что было до начала стандартизации.

2.Уровень 1.

Появление возможностей создания дерева узлов документа, а также возможность манипулирования данными.

3.Уровень 2.

Поддержка пространства имён XML и событий.

4.Уровень 3.

Состоит из шести разных стандартов:

а)DOM Level 3 Core;
б)DOM Level 3 XPath;
в)DOM Level 3 Load and Save;
г)DOM Level 3 Validation;
д)DOM Level 3 Views and Formatting;
е)DOM Level 3 Requirements.

1.2Анализ отечественных исследований методологий управления объектной моделью веб-документа

Представление того, что представляет собой DOM-дерево, и знание того, как оно компилирует ваш код является малой частью к абсолютному пониманию, как нужно манипулировать веб-страницами. Разработчику необходимо много практиковаться, что познать полное представление работы с DOM-деревом. Чтобы разобраться со всем, необходимо изучить немалое количество различных манипуляций. При разработке элементов DOM-дерева, данный элемент незамедлительно отобразится в браузере клиента без перезагрузки страницы. Удалив какой-нибудь текст, он сразу же исчезнет с экранов пользователя. Программисты пользуются возможностью управлять интерфейсом и взаимодействовать с ним через DOM, что предоставляет им большую гибкость при разработке продукта.

DOM является спецификацией, которая никак не зависит от языка. Поэтому появляется возможность использовать ее вместе со многими современными языками программирования. W3C классифицирует для DOM большой свод правил, благодаря которым можно связать ее с разными языками. Правила гласят, что существует разрешения применения DOM в определенном языке.

DOM не представляет собой язык программирования, но без JavaScript он не знал бы никакой информации о веб-страницы и содержащих в ней документов и не имел бы представления ни о какой модели. Каждый в отдельности элемент в документе –это части объектной документной модели для этого документа, поэтому всеми ими можно манипулировать с помощью DOM и скриптового языка.

JavaScript – язык сценариев, который представляет собой код, в котором хранятся наборы инструкций, которые не нуждаются в заблаговременной обработке перед запуском[10]. Во время обработки веб-страницы движком браузера выполняется компиляция кода JavaScript. Интерпретатор браузера

производит построчный анализ, обработку и исполнение исходной программы или запроса.

Когда веб-страница обрабатывается, браузер разрабатывает объектную модель на странице, которая представляет собой объектно-ориентированное изображение HTML-документа, которое обозначается в качестве интерфейса между JavaScript и самим документом и дает возможность разрабатывать динамические веб-страницы [5]:

1.JavaScript позволяет манипулировать всеми элементами и атрибутами HTML на странице.
2.JavaScript позволяет изменять все стили CSS на странице.
3.JavaScript позволяет отвечать на все существующие события на странице.
4.JavaScript позволяет создавать новые HTML-события.

С повышением встребованности и популярности JavaScript, скорость разработки и простота стали ключевыми факторами в разработки программных продуктов. Это актуализировало быстрое происхождение различных библиотек JavaScript. К необходимости расходовать дополнительные усилия для коректной работы браузеров привело различие в представлении объектной модели документа. Это стало толчком к появлению библиотек JavaScript, которые разрешали вопрос кроссбраузерного интерфейса к методам DOM.

Библиотека JavaScript –сборник классов и/или функций на языке JavaScript.

По мнению Ю.С.Медведева и В.В.Терехова [10] библиотеки JavaScriptпредоставляют возможности быстрой разработки веб-компонентов, позволяют разрешать проблемы кроссбраузерной совместимости. При этом стандартная библиотека должна содержать функции для работы со строками, датами, DOM-элементами, событиями, cookie, анимацией, запросами, и многим другим. Каждая функция возвращает значения вызывающему ее приложению, которое может в дальнейшем использовать их в зависимости от логики разработчика.

Библиотека обычно сокращает время разработки примерно на 20%, позволяя вам не беспокоиться о мелочах.

Проанализировав данные [1; 6; 8], можно сделать вывод, что самой популярнойjs-библиотекой всех времен является jQuery.

jQuery – библиотека JavaScript, основывающаяся на взаимодействии JavaScript и HTML. Она позволяет достаточно легко получать доступ к любому элементу DOM, манипулировать атрибутам и содержимым элементов DOM. Также jQuery дает возможность использовать удобный API для работы с AJAX.

Библиотека произвела переворот в программировании клиентской части веб-приложений, введя селекторы CSS для доступа к узлам DOM-дерева, обработчики событий, анимации и AJAX-запросы.

В последнее время jQuery утрачивает былую популярность, но по-прежнему остается жизнеспособным вариантом для проектов, требующих небольшого js-функционала.

ВозможностиjQuery:

1.Движок кроссбраузерных CSS-селекторов, выделившийся в отдельный проект.
2.Переход по дереву DOM, включая поддержку XPath как плагина.
3.События.
4.Визуальные эффекты.
5.AJAX-дополнения.
6.JavaScript-плагины.

Плюсы:

1.Малый размер дистрибутива.
2.Низкий порог вхождения, исчерпывающая документация в интернете.
3.Лаконичный синтаксис.
4.Легко расширяемый.

Минусы:

1.Замедляет работу приложения.
2.Может повлечь проблемы совместимости с браузером.
3.Сообщество разработчиков протестует против его повсеместного использования.

React – это библиотека JavaScript, созданная разработчиками Facebook и Instagram. Согласно опросу, Stack Overflow Survey 2017, React был признан самой

популярной технологией среди разработчиков. React также имеет честь быть самым популярным проектом JavaScript, согласно количеству звезд на GitHub.

С помощью React можно создать интерактивный интерфейс с использованием декларативного подхода, в котором вы можете контролировать состояние приложения, говоря: «Представление должно выглядеть так». Он использует компонентную модель, в которой компоненты являются повторно используемыми элементами пользовательского интерфейса, и каждый компонент имеет свое собственное состояние. Это была одна из первых библиотек, реализующих виртуальное DOM-дерево.

Согласно утверждениям А.А.Карышева [7],виртуальный DOM изображает простую копию обычного DOM. И специфической особенностью React является то, что он взаимодействует именно с виртуальным DOM, а не обычным.

При необходимости выяснить информацию о элементах, выполняется обращение к виртуальному DOM.Если требуется изменить элементы веб-страницы, то вначале изменения выполняются в виртуальный DOM, а затем новое положение виртуального DOM выполняет сравнение с текущим состоянием. И если эти состояния отличаются, то React ищет самое малое количество манипуляций, которые требуются до обновления реального DOM до нового состояния и исполняет их.

Следовательно, данная схема взаимодействия с элементами веб-страницы является гораздо быстрее и эффективнее, чем работа JavaScript с DOM напрямую.

Плюсы:

1. Компактность, эффективность, производительность и гибкость.
2. Простая модель компонентов.
3. Хорошая документация и обилие онлайн-ресурсов.
4. Возможность рендеринга на стороне сервера.
5. Растущая популярность.

Минусы:

1. Новые концепции и синтаксис, которые придется изучить.
2. Необходимы системы сборки.

3. Может требовать сторонних инструментов и библиотек.
4. Может быть несовместим с кодом и другими библиотеками, модифицирующими DOM-дерево.

Существует большое количество библиотек и фрейморков, с помощью которых можно манипулировать DOM. В своей работе А.Г.Матвеев [9] выделяет еще одну, схожую с jQuery, библиотеку Dojo.Его еще называют Dojo Toolkit. Dojo – это сборник очень полезного JavaScript-кода. Он предоставляет JavaScript-методы, с помощью которых разработчик может анимировать элементы, реализовать постепенное проявление/исчезновение элементов, осуществлять Ajax-вызовы и т.д.Главной особенностью Dojo Toolkit является простота использования и многофункциональность. С помощью данной библиотеки можно работать с DOM структурой HTML документа, а именно: находить элементы, добавлять, модифицировать.

1.3Анализ отечественных исследований методологий управления объектной моделью веб-документа

Исследование данной темы также занимаются зарубежные авторы. Одними из них является авторы статьи «Professional JavaScript Frameworks: Prototype,YUI, ExtJS, Dojo and MooTools» [24]. Согласно статье,Dojo –содержит все необходимые функции для манипуляции с DOM, хелперы событий и анимации, также AJAX функции, подобно как в jQuery. И эта библиотека развивается далее и далее.

Набор инструментов Dojo разделен на несколько основных пакетов, которые будут представлять собой полное распространение Dojo Toolkit

Эти главные пакеты:

1.dojo - изредка называемый «ядром», это главная часть Dojo, и наиболее распространенные пакеты и модули хранятся в нем. Ядро включает широкий

спектр функций, таких как AJAX, DOM-манипуляция, программирование типа, события, обещания, хранилища данных, библиотеки перетаскивания и интернационализации.
2.dijit - обширный набор виджетов (компонентов пользовательского интерфейса) и базовая система для их поддержки. Он полностью встроен в корпус Dojo.
3.dojox - собрание пакетов и модулей, которые изображают широкий спектр функций, которые основаны как на ядре Dojo, так и на Dijit. Пакеты и модули, содержащиеся в DojoX, будут иметь разную степень зрелости, обозначаемую в файлах README.
4.util - разные инструменты, с помощью которые поддерживается остальная часть инструментария, например, возможность создавать, тестировать и документировать код.

Возможности работы Dojo с DOM элементами:

1. Поиск элементов.
2. Создание DOM элементов.
3. Изменение элементов.
4. Удаление элементов.
5. Перемещение и дупликация.
6. Перебор узлов.
7. Работа с событиями.

В 2016 году большой рост популярности получил фреймворк Vue.js. График роста количества звезд Reactи Vue.jsна GitHubизображен на рисунке 3. Vue – один из самых быстроразвивающихся JS-фреймворков в 2016-м. Vue описывает себя как «Интуитивный, Быстрый и Интегрируемый MVVM для создания интерактивных интерфейсов».

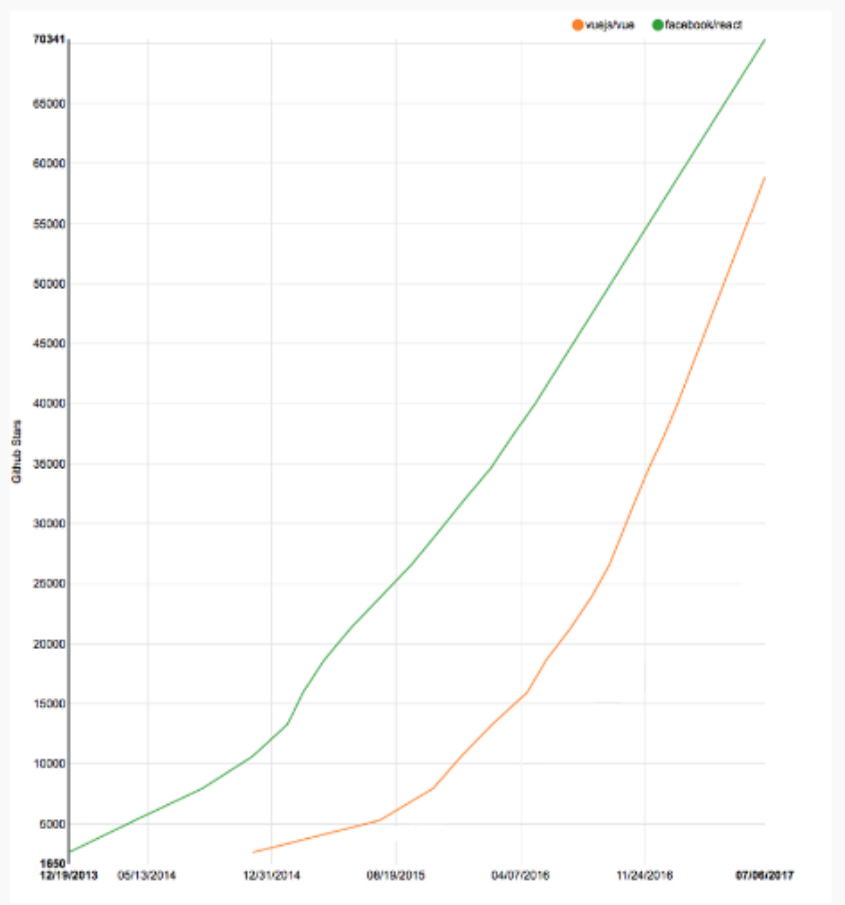

Рисунок 3 – График роста количества звезд на GitHub

Vue.js представляет современный прогрессивный фреймворк, написанный на языке JavaScript и предназначенный для создания веб-приложений клиентского уровня. Основная сфера применения данного фреймворка – это создание и организация пользовательского интерфейса. Одним из ключевых моментов в работе Vue.js является виртуальный DOM. Он так же, как и Reactпредставляет легковесную копию обычного DOM, выбирает минимальный набор компонентов, для которых надо выполнить изменения на веб-странице, чтобы реальный DOM соответствовал виртуальному. Благодаря виртуальному DOM повышается производительность приложения [5].

Плюсы:

1. Зрительские симпатии и растущая популярность;
2. Проста в освоении с нуля;
3. Мало зависимостей и хорошая производительность.

Минусы:

1.Молодой проект — высокие риски.
2.Меньше ресурсов, чем у альтернатив.

Авторы К.Свидберг и Д.Чаффер [20] пишут о том, что библиотекой JavaScriptвсех времен является jQuery. Библиотека быстро завоевала признание программистов по всему миру, благодаря упрощенной разработке веб-приложений и возможности избавиться от «синтаксической шелухи», присущей проектам на JavaScript. Она является одной из лучших библиотек, если необходимо управлять DOM или применять AJAX-запросы.Все версии jQuery полностью совместимы между собой.

Выводы по главе

В данной главе была рассмотрена основная информация о Document Object Model. Изучены ее возможности и особенности. Также изучены отечественные и зарубежные источники по теме ВКР. Были рассмотрены некоторые методологии управления DOMи выделены их краткие характеристики и возможности.

ГЛАВА 2. ИССЛЕДОВАНИЕ МЕТОДОЛОГИЙ УПРАВЛЕНИЯ DOM

2.1Объектная модель веб-документа

Объектная модель документа – это множество объектов, которые браузер создаёт в своей памяти на основе кода HTML. DOM изображает структурированное описание документа и представляет то, как с этой структурой можно работать из программ, которые позволяют изменять содержимое документа [6]. Представление DOM имеет группы узлов и объектов, которые в свою очередь имею свойства и методы. По существу, DOM соединяет веб-страницу с языками описания сценариев либо языками программирования таких страниц, которые мгновенно изменяются, если изменяется объектная модель этого документа. Таким образом, веб-разработчик при создании таких страниц работает не с HTML кодом данной страницы, а с объектами, которые браузер создаёт на основе этого кода.

Различные браузеры имеют различное представление DOM, эти реализации изображают различную степень соответствия с действительным стандартом DOM, но каждый браузер работает своим представлением DOM, чтобы позволить веб-страницам стать доступными для взаимодействия с языками сценариев.

При разработке сценария с употреблением элемента <script>, либо включая в веб-страницу руководство для загрузки скрипта, появляется возможность незамедлительно приступить к эксплуатации программного интерфейса, пользуясь элементами document или window для взаимодействия с самим документом. Программирование манипуляций с DOM может являть чем-то легким, например, вывод сообщения с употреблением функции alert() объекта window, или употребить наиболее сложные методы DOM, с помощью которых можно создавать новое наполнение, как показано в следующем примере [11]:

```
<body onload="window.alert('Привет, Мир!');">
```

В следующем примере внутри элемента <script>написан код JavaScript, данный код определяет функцию при загрузке документа. Эта функция формирует новый элемент h1, добавляет текст в данный элемент, а затем добавляет h1 в дерево документа:

```
<html>
<head>
<script>
   // запуск данной функции при загрузке документа
   window.onload = function() {
   // создание нескольких элементов впустойHTMLстранице
heading = document.createElement("h1");
      heading_text = document.createTextNode("Big Head!");
      heading.appendChild(heading_text);
      document.body.appendChild(heading);
}
</script>
</head>
<body>
</body>
</html>
```

Существуют разные типы данных, которые используются в API:

1.document

Когда возвращается объект типа document то, этот объект document является собственным корневым объектом.

2.element

Определяет элемент или узел, возвращаемый DOM API. Вместо того, чтобы утверждать, что метод document.createElement() возвращает ссылку на node,

необходимо уточнить, что элемент возвращает element, который создан в DOM. Объекты element реализуют DOM element интерфейс.

3.NodeList

Массив элементов, напоминающий массив, которые возвращается методом Document.getElementsByTagName(). Определенные элементы в массиве доступны по индексу двумя способами:

a)list.item(1)
b)list[1]

Эти способы одинаковы.

4.attribute

Когда attribute возвращается членом API (например, метод createAttribute()) – это будет ссылка на объект, который изображает специальный интерфейс для атрибутов.

5.namedNodeMap

namedNodeMap схожий с массивом, но доступ к элементу осуществляется по имени или индексу. Если элемент доступен по индексу, то это применилось для удобства перечисления. Этот тип данных использует метод item() для этих целей и также существует возможность добавлять и удалять элементы из namedNodeMap.

Document, window –это объекты, чьи интерфейсы, как правило, очень часто употребляются в программировании DOM. Выражаясьобычными словами, объект window изображает что-то в виде браузера, а объект document –ядро самого документа. Element наследуется от интерфейса Node, и эти интерфейсы вместе дают много методов и свойств, которые можно употребить у отдельных элементов. Эти элементы также могут содержатьсвои интерфейсы для взаимодействия с типами данных, которые эти элементы содержат, как в примере с объектом table в предыдущем случае [11].

Ниже изображен не большой список распространённых членов API, употребляемых в программировании веб-страниц и XML-страниц с употреблением DOM:

1.document.getElementById(id).
2.document.getElementsByTagName(name).
3.element.setAttribute.
4.window.scrollTo.
5.parentNode.appendChild(node).
6.element.innerHTML.
7.element.style.left.
8.element.getAttribute.
9.window.onload.
10.window.dump.

2.2Исследование возможностей методологий управления DOM

Представление того, что представляет собой DOM-дерево, и знание того, как оно компилирует ваш код является малой частью к абсолютному пониманию, как нужно манипулировать веб-страницами. Разработчику необходимо много практиковаться, что познать полное представление работы с DOM-деревом. Чтобы разобраться со всем, необходимо изучить немалое количество различных манипуляций. При разработке элементов DOM-дерева, данный элемент незамедлительно отобразится в браузере клиента без перезагрузки страницы. Удалив какой-нибудь текст, он сразу же исчезнет с экранов пользователя. Программисты пользуются возможностью управлять интерфейсом и взаимодействовать с ним через DOM, что предоставляет им большую гибкость при разработке продукта.

В декабре 1995 года программистом Brendan Eich был разработан язык, который в дальнейшем стал называться JavaScript. За короткое время он был переименован в JavaScript, но основным названием является ECMAScript [10].

JavaScript – язык сценариев, или скриптов. Скрипт демонстрирует собой программный код, в котором содержатся наборы инструкций, которые не нуждаются в заблаговременной обработке перед запуском[10]. Во время обработки веб-страницы движком браузера выполняется обработка код JavaScript Интерпретатор браузера производит построчный анализ, обработку и исполнение исходной программы или запроса. Он поддерживает встроенные объекты, а также дает возможность создавать или удалять свои собственные. Объекты могут наследовать свойства непосредственно друг от друга, образуя цепочку объект-прототип. JavaScript дает возможность манипулировать элементами и атрибутами HTML на странице, изменять все стили CSS на странице.

С помощью JavaScript можно:

1.Преображать веб-страницы динамически.
2.Перед отправкой на сервер делать проверку данных форм и многое другое.
3.Обрабатывать код через определенные промежутки времени.
4.Манипулировать поведением браузера (открывать новые окна, загружать указанные документы и т.д.).
5.Разрабатывать и считывать cookies.
6.Определять, какой браузер использует пользователь.

Для наглядности приведем наиболее часто используемые операции при разработке анимированных веб-страниц на чистом JavaScript[8].

1.Поиск элемента по идентификатору.

Без поиска не обходится ни одна веб-страница. Всем известно, что поиск по id наиболее приемлемый, и употребляют его. Для поиска употребляется следующий код:

```
document.getElementById('id');
```

2.Поиск элементов по классу.

Зачастую поиском по идентификатору разработчики не ограничиваются, поэтому необходимо использовать поиск элементов другим образом. Например, поиск по классу:

```
document.getElementsByClassName('class');
```

3.Добавление элемента

Каждый разработчик должен знать, каким образом можно добавить элементы на страницу. Для примера применилось добавление однотипных span непосредственно к body:

```
varspn = document.createElement('span');
spn.setAttribute('class','testspan');
document.body.appendChild(spn);
```

4.Изменение класса элемента.

Зачастую определять класс не требуется – нужно добавить его, или удалить. На нативном JavaScriptкод для этого получается довольно объемным:

```
varclasses = nElement.className.split(' ');
var ind = classes.indexOf('testToggle');
if(ind==-1) classes.push('testToggle');
else classes.splice(ind,1);
nElement.className = classes.join(" ");
```

5.Изменения стиля элемента.

Предположим, что необходимо с помощью чистого языкаJavaScript изменить цвета фона элемента списка (<li>) в неупорядоченный список выше.

Возьмем, к примеру, следующий HTML:

```
<div class = "container">
    <ul class = "list">
        <li> Текст здесь </ li>
        <li> Текст здесь </ li>
        <li> Текст здесь </ li>
        <li> Текст здесь </ li>
        <li> Текст здесь </ li>
    </ul>
```

```
</div>
```

Используя чистый JavaScript, код будет выглядеть примерно так:

```
var myListCollection = document.getElementsByTagName("ul");
for (var i = 0; i < myListCollection.length; i++) {
if (myListCollection[i].className === "list") {
myListCollection[i].childNodes[0].style.backgroundColor = "blue";
        }
 }
```

Со временем известность JavaScript возрастала, легкость разработки динамических элементов интерфейса стала иметь значение ключевой роли в веб-разработке. Это стало толчком для появления библиотек JavaScript, которые разрешали вопрос кроссбраузерного интерфейса к методам DOM.

По мнению Ю.С.Медведева и В.В.Терехова [4] библиотекаJavaScript библиотека должна содержать функции для работы со строками, датами, DOM-элементами, событиями, cookie, анимацией, запросами, и многим другим.

Исследовав данные [1; 2; 3], можно сделать вывод, что самой популярной js-библиотекой является jQuery.

jQuery - это библиотека, которая в большей части облегчает и ускоряет разработку JavaScript скриптов.

Слоган jQuery «write less, do more», что обозначает – «пиши меньше, делай больше», отображает ее главное предназначение [13].

jQuery предоставляет возможность разрабатывать анимацию, в большой степени упрощает выбор элементов и создание AJAX запросов.

Данная библиотека трудится и работает со всеми браузерами (IE 6.0+, FF 2.0+, Safari 3.0+, Opera 9.0+, Chrome). Это значит, что нет необходимости думать о кроссбраузерной совместимости JavaScript кода.

Для jQuery придумано и разработано большое количество плагинов, которые еще больше повышают ее возможности. Библиотека произвела переворот в программировании клиентской части веб-приложений, введя селекторы CSS для доступа к узлам DOM-дерева, обработчики событий, анимации и AJAX -запросы.

jQuery, на самом деле, не что иное, как JavaScript. Весь код, написанный в jQuery, преобразуется в JavaScript. Одна строка кода jQuery, может быть равна многим строкам кода чистого JavaScript. Благодаря этому разработчику необходимо написать только маленькие строки кода.

Для начала работы с jQuery на вашей странице, необходимо включить одну строку кода в заголовок вашей страницы. Например,

<scriptsrc = "https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.2.1.min.js" >

</ script>

Данная строчка включает библиотеку jQuery на страницу с помощью Microsoft CDN (Сеть доставки контента). Библиотека jQuery представляет собой один файл с кодом JavaScript.

Приведм пример для нативного JavaScript наиболее часто используемых операции при создании анимированных веб-страниц, но уже с помощью jQuery [8].

1.Поиск элемента по идентификатору.

Для поиска элемента код на jQuerybудет значительно меньше:

$('#id');

2.Поиск элементов по классу.

Поиск по классу:

$('.class');

3.Добавление элемента.

Для этой операции код на jQueryзначительно меньше, чем на JavaScript. Для примера использовалось добавление однотипных span непосредственно к body:

$(document.body).append($('<spanclass="testspan">'));

4.Изменение класса элемента.

Зачастую определять класс не требуется – нужно добавить его, или удалить. На jQuery код гораздо меньше:

```
jElement.toggleClass('testToggle');
```

5.Изменения стиля элемента.

Для примера на jQuery возьмем такой же HTML-документ [13]:

```
<div class = "container">
    <ul class = "list">
<li> Текст здесь </li>
<li> Текст здесь </li>
<li> Текст здесь </li>
<li> Текст здесь </li>
    <li> Текст здесь </li>
    </ul>
</div>
```

Используя jQuery, код будет выглядеть примерно так:

```
$("ul.list li:first-child").css("background-color", "blue"); }
```

При разработке это значительно занимает меньше времени, а также в раз уменьшается объем кода.

Следовательно, можно сделать вывод, что библиотека jQuery, благодаря своим возможностям, сокращает время разработки примерно на 20%, позволяя не беспокоиться о мелочах.

2.3Результаты исследования возможностей методологий управления DOM

Исходя из проведенного исследования, jQuery подходит для приложений, которые требуют быстрого развития. jQuery заботится об общих ошибках браузера, гладя исправления прямо в библиотеке. jQuery также заботится о проблеме совместимости браузера, которая является проблемой для разработчика во время развертывания.

Использование JavaScript или jQuery действительно зависит от потребностей и других факторов. Большое количество проектов веб-разработки будут работать отлично, используя jQuery.

В рамках научно-исследовательской работы было проведено исследование методологий и их возможностей относительно управления объектной моделью веб-документа. В сводной таблице 1 приводятся результаты исследования.

Таблица 1 – Результаты исследования методологий управления DOM

JavaScript	jQuery
Слабо типизированный язык динамического программирования	Быстрая и сжатая библиотека JavaScript
Язык сценариев взаимодействия интерфейса и управления содержимым документа	Библиотека, которая упрощает и ускоряет обработку событий, анимацию и Ajax
Интерпретируемый язык	Использует ресурсы, предоставленные JavaScript, чтобы упростить работу
Нужно писать собственные сценарии, которые могут занять время	Не нужно писать много скриптов, которые уже существуют в jQuery
Разработчики должны обрабатывать совместимость с несколькими браузерами, написав собственный код JavaScript	Является многосерверной библиотекой JavaScript, которая уменьшает работу разработчиков во время развертывания
Разработчики склонны делать много распространенных ошибок, связанных с браузером	Разработчикам не нужно беспокоиться о проблемах совместимости браузеров
Больше строк кода	Меньше строк кода

Выводы по главе

В данной главе рассмотрены возможности методологий управления DOM, таких как JavaScriptи jQuery. Выделены основные характеристики данных методологий и приведены и сводной таблице.

ГЛАВА 3. ТЕОРЕТИЧЕСКОЕ СРАВНЕНИЕ JQUERY, REACT И JAVASCRIPT

3.1jQuery. "write less, do more"

Со временем легкость разработки динамических элементов пользовательского интерфейса стала иметь значение ключевой роли в веб-разработке. Это стало толчком для появления библиотек JavaScript, которые разрешали вопрос кроссбраузерных интерфейсов к методам DOM.

Исследовав данные [1; 2; 3; 5] можно сделать вывод, что самой популярной js-библиотекой всех времен является jQuery.

jQuery – это библиотека, которая в большей степени облегчает и ускоряет написание JavaScript кода. Логотип библиотеки изображен на рисунке 4.

Рисунок 4 – Логотип библиотеки jQuery

jQuery однозначно стал стандартом в веб-разработки. Есть много замечательных js-фреймворков, которые удостаиваются внимания, но jQuery удивил всех своей легкостью, изящностью, магией. Разработчики программируют с jQuery, создают плагины для jQuery, даже пишут статьи про jQuery, но мало кто знает, как построен jQuery.

Библиотека взаимодействует со всеми браузерами (IE 6.0+, FF 2.0+, Safari 3.0+, Opera 9.0+, Chrome). Данная особенность дает возможность разработчикам больше не беспокоиться о кроссбраузерной совместимости JavaScript кода.

С помощью библиотеки jQuery:

1.Абстрагируется интерфейс объектной модели документа.
2.Выравниваются все различия реализаций DOM, существующие между браузерами.
3.Исправляются известные браузерные баги, связанные с CSS и DOM.

Все это собирается в более простой и менее корявый API, нежели обычный API DOM.

В работе с jQuery есть много полезного. Например, задачи, которые решаются с помощью выполнения множества операций, благодаря библиотеке гораздо упрощаются. Разработчики выбирают jQuery по следующим причинам [2]:

1.jQuery позволяет добиться большего результата при меньшем объеме кода.
2.Методы jQuery применимы к целым группам элементов. Предлагаемый в DOM-модели стандартный подход, основанный на шаблонной цепочке действий «выбрать-повторить-изменить», больше не нужен. Следствием этого является уменьшение количества циклов «for» в коде, что способствует снижению вероятности появления в нем ошибок.
3.Библиотека jQuery справляется с различиями в представлении DOM в различных браузерах.
4.Библиотека jQuery имеет открытый исходный код. Программист, в случае если результат ожидаемый результат не совпадет с действительным, то он сможет свободно обратиться к коду библиотеки на JavaScript и, если это потребуется, внести свои необходимые изменения.

Для начала работы с jQuery существует два способа подключения библиотеки.

Способ 1. Скачать в папку и подключить файл.

Нужно зайти на официальный сайт библиотеки. Там предоставляются две версии jQuery: production для работы уже с готовыми приложениями и development непосредственно для разработки. Версия для разработки имеет комментарии и организованный код, но его объем около 276 КБ. В сокращенной

версии нет комментариев и код в ней не структурирован, но зато ее объем около 88 КБ, поэтому разработанные продукты с ней будут обрабатываться намного быстрее. После выбора необходимой версии требуется скачать файл.

Чтобы добавить файл к проекту нужно в секцию head страницы пропечать следующий код:

<script type="text/javascript" src="путь_к_скачанному_файлу/jquery.js">

</script>

Способ 2. Подключить jQuery напрямую с CDN.

Данный способ позволяет подключить библиотеку с сайта, не занимая место на жестком диске. Особенно это актуально при большом количестве маленьких проектов и для обучения.

Разработчики рекомендуют подключать библиотеку от Google Developers. Необходимо скопировать строку и вставить ее на нашу страничку.

<scriptsrc="https://ajax.googleapis.com/ajax/libs/jquery/jquery-3.2.1.min.js">

</script>

Данная строка включает библиотеку jQuery на страницу с помощью Microsoft CDN (Сеть доставки контента).

Основной функций в библиотеке является функция jQuery()– с помощью нее выполняются основные действия на веб-странице: поиск, манипулирование элементами. Она возвращает объект jQuery.

Для удобства и быстроты написания jQuery() сокращают до $(). В качестве аргументов данной функции передают селекторы, над которыми данная функция выполняет манипуляции.

Например, $('div') –выбирает со страницы все блоки.

Как и в JavaScript, перед началом работы со страницей, необходимо быть уверенным, что DOM загрузился, для этого в jQuery существует событие ready. Используя событие ready и помещая в него вызовы функций, вы всегда будете уверены, что функции будут выполнены только после загрузки страницы.

$(document).ready(function(){

Вызов функций

});

Есть еще более простой способ записи вызова функций, который представляет собой сокращенный вариант предыдущей и занимает меньше объема и времени написания:

$(function(){

вызов функций

});

Селекторы – это выражения, которые используются в jQuery для поиска элементов HTML на странице. Селекторы могут находить на странице элементы по имени, по классу, по id, по вложенности и т.д. В отличие от JS, где способ работы с элементами различается для одного элемента и массива, в jQuery способы работы одинаковы и с одним элементом и с массивом.

Один из исследователей, Эли Бурштайн [12], разработал гусеничный ход, который проанализировал первые 100.000 сайтов. Он выяснилось, что 45% этих сайтов используют библиотеку JavaScript. Из них большинство веб-сайтов используют, jQuery, MooTools, Prototype и YUI. На рисунок 5 изображена разница в популярности между библиотеками JavaScripe. Другие исследования показывают аналогичные результаты, которые схожи с исследованиями Эли Бурштейна[12].

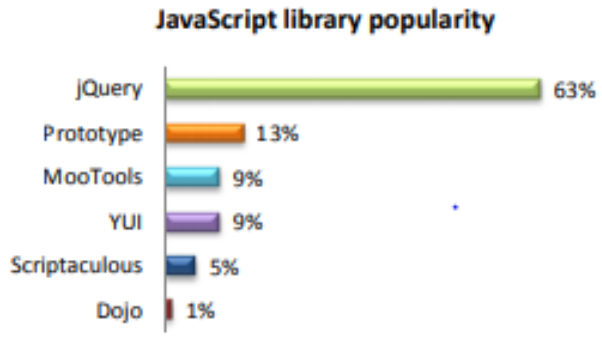

Fig. 1. JS libraries used among the Alexa top 100.000 websites.

Рисунок 5 – Результаты исследования Эли Бурштейна

Результаты данных исследований подтверждают, что в свое время, библиотека jQuery была самых популярных в мире.

3.2Библиотека React.js

React.js– это библиотека JavaScript, созданная разработчиками Facebook и Instagram, которая используется для создания пользовательского интерфейса [8]. Логотип React.js изображен на рисунке 6.

Рисунок 6 – Логотип React

Летом 2018 года в мире JavaScript-фреймворков произошло важное событие: в борьбе Vue.js vs React.js первый победил по количеству людей, которые «поставили ему звезды» на Github [9]. Хотя до этого момента React.js уверенно лидировал и оставался на высоте продолжительное время. Рейтинг за последние 5 лет изображен на рисунке 7.

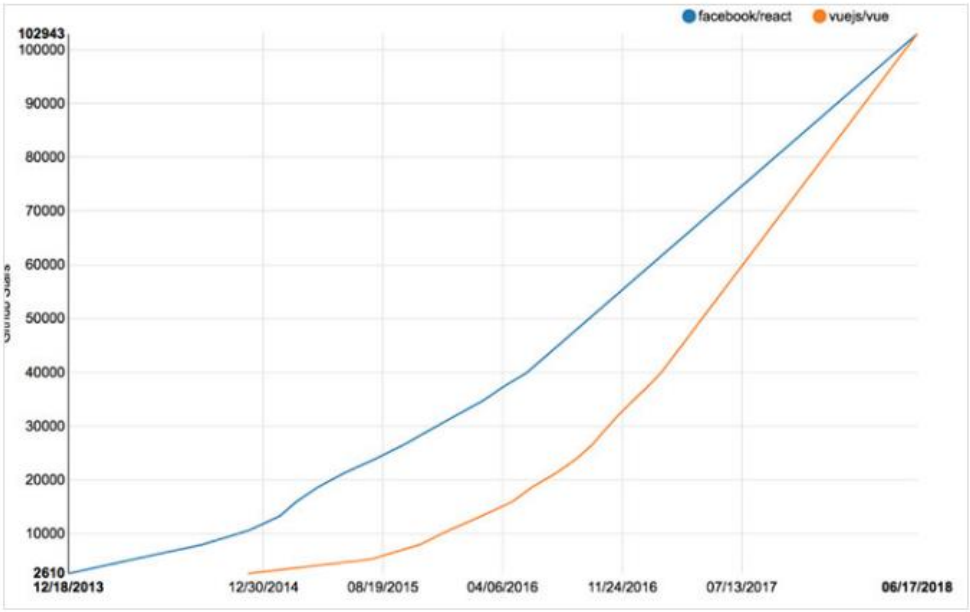

Рисунок 7 – Рейтинг «звезд» на Github

React.js представляется идеальный инструмент для разработки масштабируемых веб-приложений особенно в тех ситуациях, когда приложение

представляет SPA. Библиотека является простой в понимании, имеет понятный и лаконичный синтаксис. Это была одна из первых библиотек, реализующих Virtual DOM-дерево[8].

Согласно утверждениям А.А.Карышева [4], виртуальный DOM изображает обычную копию простого DOM. Специфической особенностью React.js является то, что данная библиотека взаимодействует именно с Virtual DOM, а не обычным.

Для решения проблемы производительности появилась концепция виртуального DOM, с которым работает React.js.

Если приложению необходимо выяснит информацию об элементах, то выполняется обращение к виртуальному DOM. Если требуется изменить элементы веб-страницы, то изменения вначале вводятся в виртуальный DOM[17]. Потом изменения виртуального сравнивается с текущим состоянием обычного DOM. И если эти состояния отличаются, то React.js ищет наименьшее количество манипуляций, которые требуются до обновления реального DOM до нового состояния и исполняет их. С Virtual DOM программисты могут достичь наивысшей производительности своего продукта.

Следовательно, данное взаимодействие с элементами веб-страницы работает наиболее быстро и эффективнее, чем прямая работа из JavaScript с DOM.

Опрос, проведенный среди разработчиков сайтом StackOverflow дал понять, какая технология стала наиболее популярной [10]. Более 50 тысяч разработчиков рассказали информацию о своей работе и привычках, которые они применяют в разработке. Библиотека React.js стала одной из любимой и используемых технологий, а также самой необходимой технологией на StackOverflow. Результаты опроса изображены на рисунке 8.

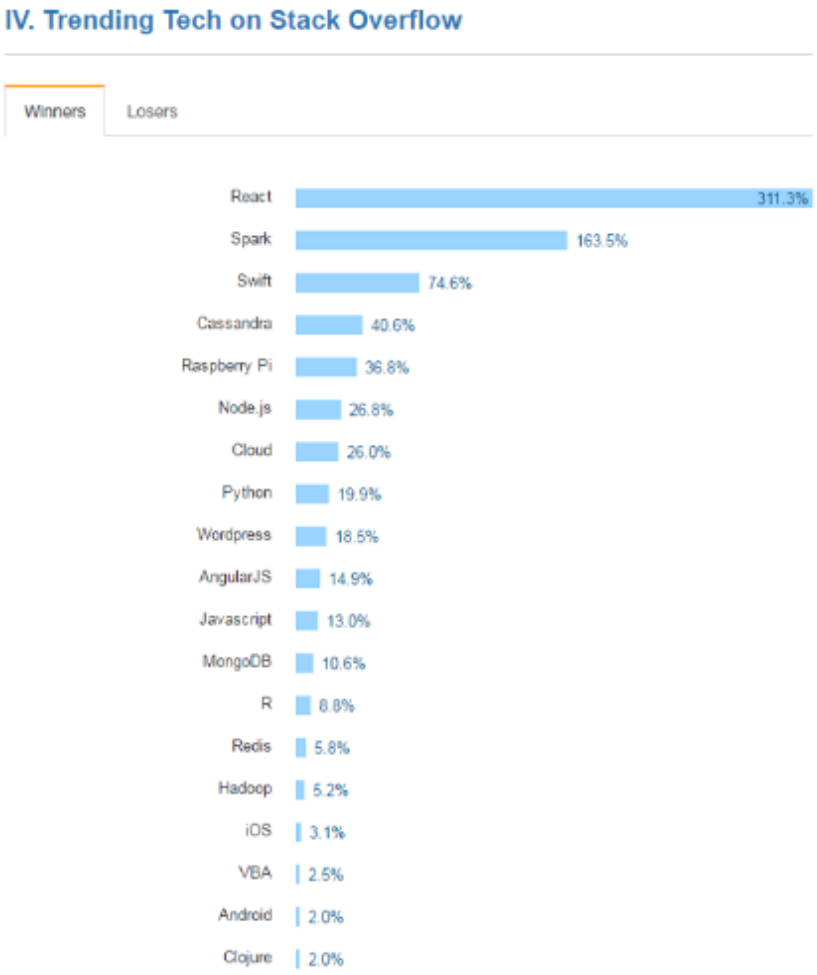

Рисунок 8 – Результат опроса на StackOverflow

Одной из отличительных особенностей React.js является возможность употреблять JSX, который так близок по синтаксису с HTML, который обрабатывается в JavaScript. С React.js разработчик может разрабатывать изоморфные приложения, которые избавляют от ситуации, когда пользователь долгое время ждет завершения загрузка данных и на экране высветится что-то кроме анимации загрузки. Разработанные компоненты могут изменены и использованы заново в новых проектах.

Концепции React.js:

Элементами JavaScript являются объекты, которые представляют HTML-элементы. Их не существуют в браузере. Они обрисовывают DOM-элементы, такие как h1, div, или section.

Компоненты — это элементы React, разработанные программистом. Обычно это части хранят свою структуру и функциональность. Например, такие как NavBar, LikeButton, или ImageUploader.

JSX–это техника формирования элементов и компонентов React. С JSX затрачивается гораздо меньше усилий, он компилируется в JavaScript перед запуском в браузере.

Virtual DOM –это дерево React элементов на JavaScript. React обрисовывает виртуальный DOM в браузере, чтоб сделать интерфейс видимым. React наблюдает за изменениями в виртуальном DOM и автоматически делает изменения DOM в браузере так, чтоб он соответствовал виртуальному.

Основными преимуществами использования React.js для разработчика станут[18]:

1.Каждый компонент будет обрисован, глядя на исходных код.
2.Виртуальный DOM может увеличить производительность приложений.
3.Употребление изоморфного подхода помогает выполнять рендеринг страниц быстрее, что помогает сократить ожидание пользователя при загрузке страницы или приложения. В результате время разработки и затраты уменьшаются.
4.Переиспользование кода позволяет проще создавать мобильные приложения. Код, написанный во время разработки сайта, может быть употреблен для создания мобильного приложения. Если в планы разработчика входит не только создание сайта, но и мобильного приложения, тогда разработчику нет необходимости искать людей для создание мобильного приложения.

Так же, как и для jQuery для установки React.js существует несколько способов [14]:

1.Для одностраничного приложения.

а) Следует использовать Create React App и npm:

```
npm install -g create-react-app
create-react-app hello-world
cd hello-world
npm start
```

б) Добавление React к уже существующему приложению:

npminstall --savereact

в) Использование CDN:

<script crossorigin src="https://unpkg.com/react@16/umd/react.development.js"></script>

<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>

Указанные выше версии предназначены только для разработки и не подходят для производства.

Минимизированные и оптимизированные производственные версии React доступны по адресу:

<script crossorigin src="https://unpkg.com/react@16/umd/react.production.min.js"></script>

<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>

С каждым годом библиотека React.js стает популярнее и востребование. По статистике количество вакансий, в которых требуются разработчики, использующие React.js, превзошло количество вакансий, где требуются люди, знающие jQuery, хотя этот инструмент на протяжении долго времени оставался самым популярным и востребованным. Статистика вакансий на 2018 год изображена на рисунке 9.

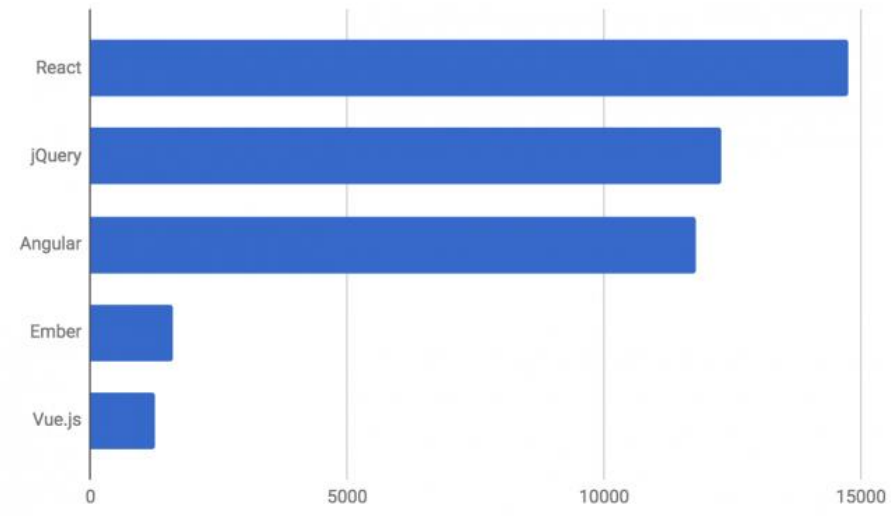

Рисунок 9 – Статистика технических вакансий за 2018 год

3.3JavaScript

JavaScript – язык сценариев, или скриптов. Скрипт демонстрирует собой программный код, в котором содержатся наборы инструкций, которые не нуждаются в заблаговременной обработке перед запуском [10]. Во время обработки веб-страницы движком браузера исполняется код JavaScript. Интерпретатор браузера производит построчный анализ, обработку и исполнение исходной программы или запроса. Логотип изображен на рисунке 10.

Рисунок 10 – Логотип JavaScript

Программисты при разработке веб-приложений часто используют JavaScript из-за некоторых ключевых особенностей языка, а именно:

1.Приведение типов данных проводится автоматически.
2.Функции выступают объектами базового класса,
3.Автоматическая очистка памяти. Так называемая, сборка мусора делает JavaScript похожим на C# или Java.

Если программиста спросить для чего нужен язык JavaScript, он с уверенность ответит, что язык позволяет «оживлять» неподвижные страницы сайтов с помощью скриптов.

Чаще всего язык используют для разработки веб-приложений и мобильных приложениях. А также принимает «активное участие» в AJAX. Эта технология дает возможность увеличить производительность приложения, выполняя обмен данными с сервером в «фоновом» режиме.

JavaScript предоставляет большое количество возможностей для решения самых разнообразных задач. Библиотека дает возможность употреблять большинство шаблонов программирования применительно к конкретным условиям. Востребованность JavaScript представляет перед разработчиком огромное количество библиотек, которые позволяют значительно упростить написание кода и привести к нулю несовершенства синтаксиса. Однако написание кода на нативном JavaScript является долго и тягостной работой для программиста.

Программируя с JavaScript нужно писать свои собственные сценарии, которые отнимают много времени. Так как у него нет написанных. А также С помощью JavaScript необходимо обрабатывать совместимость с несколькими браузерами, написав собственный код.

По сравнению с библиотеками, которые были исследованы выше, JavaScript разработчику не нужно импортировать и добавлять библиотеки для запуска кода. Код просто должен быть записан внутри тега <script> и </ script> в HTML.

Еще одним способом является подключение отдельного файл со скриптом, следующим образом:

<script src="/my/script.js"></script>

При этом файл /my/script.js содержит javascript-код, который иначе мог бы находиться внутри тега <script>.

Это очень удобно, если один и тот же файл со скриптами необходимо подключать на разных страницах.

3.4 Результаты исследований jQuery, React.js и JavaScript

Исследовав разные источники и литературу для дальнейшего сравнения были выбраны следующие критерии: тип, язык, простота использования, совместимость, длина кода, скорость, клиентский/серверный язык, объем файлов, способ подключения, синтаксис. Данные представлены в таблице 2.

Таблица 2 – СравнениеjQuery, React.js и JavaScript

Критерий	jQuery	React.js	JavaScript
Тип	Библиотека JavaScript	Библиотека JavaScript	Язык программирования.
Язык	Использует ресурсы, предоставляемые JavaScript, чтобы упростить процесс.	Использует ресурсы, предоставляемые JavaScript, чтобы упростить процесс.	Написан на C. Это интерпретируемый язык.
Простота использования	Не нужно много писать, так как сценарии уже существуют	Не нужно много писать, так как сценарии уже существуют	Необходимо писать свои собственные сценарии, которые отнимают много времени

Продолжение таблицы 2.

Совместимость	Кроссплатформенный	Кроссплатформенный	Необходимо обрабатывать совместимость с несколькими браузерами, написав собственный код
Длина кода	Благодаря сценариям сокращается написание кода	Благодаря сценариям сокращается написание кода	Код длинный и громоздкий
Скорость обработки	Быстро работает с современными браузерами и компьютерами. Подходит для сложных операций	Работает с VirtualDOM, что сокращает время обработки, увеличивая скорость обработки	Чистый JavaScript быстрее получает доступ к DOM, так как сокращает накладные расходы, которые имеет JQuery
Клиентский/ серверный язык	Обычно используется на стороне клиента.	Обычно используется на стороне клиента.	Обычно используется на стороне клиента
Объем файлов	Легкий по сравнению с другими библиотеками JavaScript	Легкий по сравнению с другими библиотеками JavaScript	Помимо необработанного JavaScript, он тяжелее, чем JQuery

Продолжение таблицы 2.

Способ подключения	1. Может вставлен в теги <script> и </ script> внутри HTML. 2. Загрузки с сайта jQuery.com 3. Включена из CDN.	1. Для одностраничного приложения Следует использовать Create React App и npm: npm install -g create-react-app cdhello-world npm start 2. Добавление React к уже существующему приложению: npm install --save react 3.Использование CDN	1. Код необходимо писать внутри тега <script> и </ script> в HTML. Не нужно импортировать/добавлять библиотеки. 2. Подключение отдельного файл со скриптом, следующим образом: <script src="/my/script.js" >
Синтаксис	$ - знак для определения jQuery. селектор - запрос для поиска элементов HTML. action - действие JQuery, которое нужно выполнить.	Применяется синтаксис JSX, который воспринимается как обычный HTML	Не существует специальных символов JavaScript. Мы можем просто начать писать код JavaScript в теге script в HTML
Синтаксис	$ - знак для определения jQuery. селектор - запрос для поиска элементов HTML. action - действие JQuery, которое нужно выполнить.	Применяется синтаксис JSX, который воспринимается как обычный HTML	Не существует специальных символов JavaScript. Мы можем просто начать писать код JavaScript в теге script в HTML

Исходя из вышеизложенной таблицы, для практического сравнения выбраны следующие критерии:

1. Простота использования.
2. Совместимость.
3. Длина кода.
4. Скорость написания и обработки кода.
5. Объем файлов

ГЛАВА 4. ПРАКТИЧЕСКОЕ СРАВНЕНИЕ JQUERY, REACT И JAVASCRIPT

4.1ToDo-приложение

Для проведения сравнительного анализа методологий управления объектной моделью веб-документа были разработаны три ToDo-приложений, использующие три рассмотренных ранее библиотек и фреймворков для написания скриптов.

«ToDo» – это приложение, напоминающее «Hello World» для фреймворков. Это базовое приложение, показывающее основную функциональность библиотеки.

Данные приложения имеют в своей основе одинаковую разметку на языке HTML и визуальное оформление на языке CSS. В качестве библиотек, отвечающих за работу с объектной моделью документа и обновлением ее состояния, используются библиотека jQuery, библиотека React и JavaScript.

Функционал приложений был выбран одинаковым, чтобы можно было провести анализ и сравнение на основе выбранных критериев:

1. Простота развертывания.
2. Простота использования.
3. Совместимость.
4. Длина кода.
5. Скорость обработки кода.
6. Объем файлов.

4.2Действия в приложении, необходимые для проведения анализа

Для рассмотрения основных возможностей и проведения сравнительного анализа, был выделен ряд функциональных особенностей приложения, которые

необходимо реализовать с использованием jQuery, React и JavaScript соответственно.

Приложение заключается в том, что есть 3 типа блоков: картинка, стиль и рисовалка. Каждый блок выполняет свою функцию. Блок «Картинка» отображает картинку по url-адресу. В блоке «Стиль» можно изменять параметры блока, меняя его CSS. В последнем блоке можно рисовать карандашом на белом фоне. В каждом блоке отображается время его создания. Такие блоки можно создавать и удалять.

Также существует панель управления, в которой можно указать количество блоков, которые необходимо создать, по формуле: 3*n, где n-количество, вводимое в панель управления. В панели управления также отображается время создания всей страницы и среднее время создание одного блока.

К вычисляемым значениям относятся количество создаваемых блоков, время создания страницы и время создание каждого блока в отдельности и среднее время создание одного блока.

4.3Интерфейс приложения

После определения функционала, можно приступить к проектированию интерфейса приложения, который будет использован для создания отдельных компонентов приложения.

В интерфейсе приложения будет представлена панель управления с окном для ввода количества создаваемых блоков и отображением времени создания страницы, а также блоки приложения. Блок «Картинка» с окном для ввода url-адреса картинки, блок «Стиль» с окном отображения стиля и блок «Рисовалка» с окном для рисования карандашом. Также на каждом блоке будет отображаться время создание каждого блока и функция для удаления.

Интерфейс приложения, реализованного при помощи jQuery, представлен на рисунке 11.

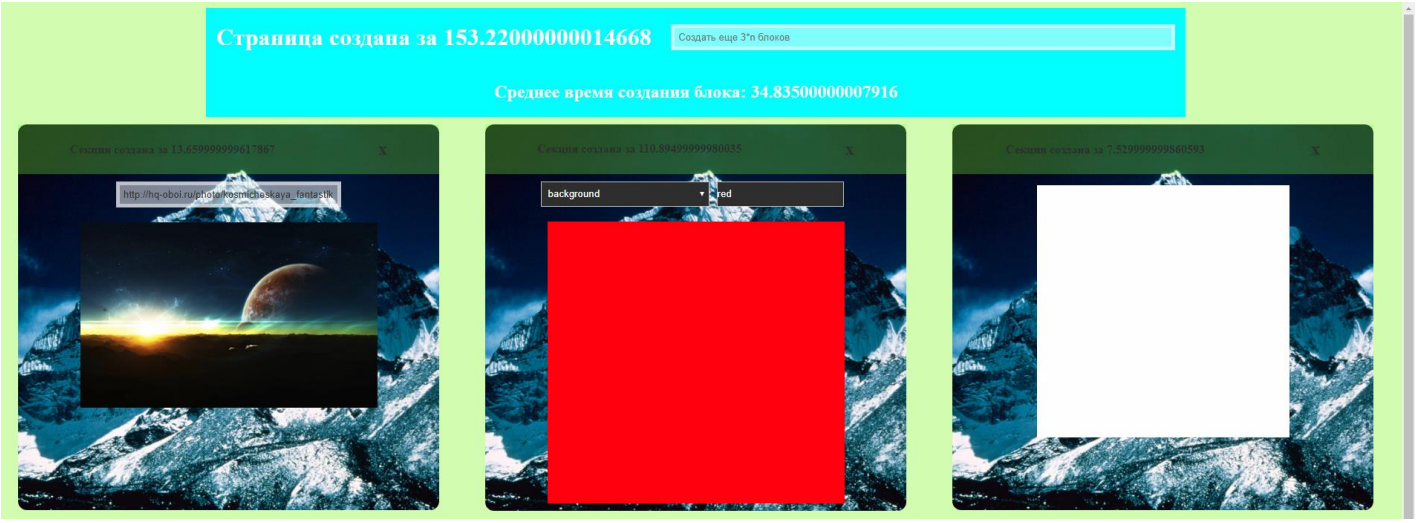

Рисунок 11 – Интерфейс ToDo-приложения

4.4Простота развертывания

Каждое приложение имеют в своей основе одинаковую разметку на языке HTML и визуальное оформление на языке CSS. Разметка HTML приложения на jQuery представлена на рисунке 12. В качестве иконок выбраны логотипы jQuery, React и JavaScript.

```
<!DOCTYPE html>
<html lang="RU-ru">
<head>
    <meta charset="UTF-8">
    <title>Jquery приложение</title>
    <link rel="shortcut icon" href="https://im0-tub-ru.yandex.net/i?id=c95c56fab9fbb7b07bdd829d78038f9e-l&n=13"
    type="image/x-icon">
    <script type="text/javascript" src="jquery-3.3.1.min.js"></script>
    <script type="text/javascript" src="all.min.js"></script>
</head>
<body></body>
</html>
```

Рисунок 12 – Размета HTMLToDo-приложения на jQuery

В качестве иконок выбраны логотипы jQuery, React и JavaScript. Ниже подключены файл с библиотекой jQuery, который был скачен с официального сайта и файл написанного скрипта.

<script type="text/javascript" src="JQuery-3.3.1.min.js"></script>

<script type="text/javascript" src="all.min.js"></script>

Для приложения на React ничего скачивать не требуется, необходимо использовать Create React App и npm:

npminstall -g

npmstart

Для нативного JavaScript ничего скачивать и подключать стороннего не требуется, необходимо лишь подключение написанного на JavaScript скрипта.

<script type="text/javascript" src="all.min.js"></script>

4.5Совместимость

jQuery и React являются кроссплатформенными. Приложения, написанные с помощью этих двух библиотек, являются кроссбраузерными, что позволяет в большинстве случаев поддерживаться всеми браузерами.

JavaScript не является кроссплатформенным языком программирования, поэтому для него необходимо обрабатывать совместимость с несколькими браузерами, написав собственный код, что занимает определенное время у разработчика. В данном случае рассчитано, что приложение будет поддерживаться всеми новыми браузерами.

4.6Длина кода

Для точного сравнения по данному критерию было принято, написать одинаковый функционал для все трех приложений.

Визуальное оформление на языке CSS для каждого приложения одинаковое и составляет 143 строчки кода. Разница заключается в том, что в отличии от jQuery и JavaScript, где CSS прописан в скрипте, то в React стили прописаны в файле формата .json и импортируется в главный скрипт следующим образом:

import defData from './data.json';

На примере блока «Картинка» можно увидеть наглядное отличие в длине кода.

Для приложения на jQuery блок «Картинка» занимает 14 строк. Для приложения на React 15 строк. Для приложения на нативном JavaScript 40 строк. Примеры кода представлены на рисунках 13, 14 и 15 соответственно.

```
166     case 'img':
167         section.append(`<div id="sd${id}"><input id="si${id}"/><img id="sim${id}" src="#" alt="Не удалось загр
168         let si = $(`#si${id}`),
169             sim = $(`#sim${id}`);
170         $(`#sd${id}`).css(defData.blocks.img.style);
171         si.css(defData.blocks.img.innerStyle.input);
172         sim.css(defData.blocks.img.innerStyle.img);
173 
174         si.change(function () {
175             sim.attr('src', si.val());
176         });
177         si.val(defData.blocks.img.data.value);
178         si.change();
179         break;
```

Рисунок 13 – Пример кода ToDo-приложения на jQuery для блока «Картинка»

```
14              case 'img':
15                  block = (
16                      <div style={this.props.defData.blocks.img.style}>
17                          <input style={this.props.defData.blocks.img.innerStyle.input}
18                                 value={this.state.ingUrl}
19                                 onChange={(val) => {
20                              this.setterState.bind(this, 'ingUrl', val)
21                          }}/>
22                          <img
23                              style={this.props.defData.blocks.img.innerStyle.img}
24                              alt="Не удалось загрузить"
25                              src={this.state.ingUrl}/>
26                      </div>
27                  );
28                  break;
```

Рисунок 14 – Пример кода ToDo-приложения на React для блока «Картинка»

```
403     imgBlock ({
404         block = undefined,
405         value = undefined
406     }) {
407         if (block) {
408             let doAfter;
409             creater({
410                 container: block,
411                 children: [
412                     new Block({
413                         name: 'input',
414                         attrs: {
415                             type: 'text',
416                             placeholder: 'Введите url ктинки',
417                             style: createStyle(defData.blocks.img.innerStyle.input)
418                         },
419                         callback: (block) => {
420                             block.addEventListener('change', (ev) => {
421                                 if (block.parentElement)
422                                     block.parentElement.dataset.img = block.value;
423                             });
424                             block.value = value;
425                             doAfter = () => {
426                                 block.dispatchEvent(new Event('change'));
427                             }
428                         }
429                     }).block,
430                     new Block({
431                         name: 'img',
432                         attrs: {
433                             src: '#',
434                             placeholder: 'Не удалось загрузить картинку',
435                             style: createStyle(defData.blocks.img.innerStyle.img)
436                         }
437                     }).block
438                 ]
439             });
440
441             doAfter();
442         }
443     }
```

Рисунок 15 – Пример кода ToDo-приложения на JavaScript для блока «Картинка»

Общая длин кода для скрипта ToDo-приложения с помощью jQuery составило 268 строк, для приложения с помощью нативного JavaScript одним файлом составило 753 строки. Скрипт для приложения, написанного с помощью React включает в себя несколько файлов и суммарное количество строк составило около 250.

4.7Скорость обработки кода

Для точного сравнения по данному критерию было принято сделать таймер, который будет отсчитывать время создания всей страницы целиком, каждого блока в отдельности и вычислять среднее время создания одного блока.

Использовалось вычисление времени с помощью метода performance.now(), который возвращает временную меткуDOMHighResTimeStamp, измеряемую в миллисекундах, с точностью до одной тысячной миллисекунды. Все цифры для наглядности были представлены в интерфейсе приложений.

Чтобы сравнить время обработки, для примера создадим в приложении 3*n блоков, где n=1000.

Результаты приложений представлена на рисунках 16, 17 и 18 и таблице 3.

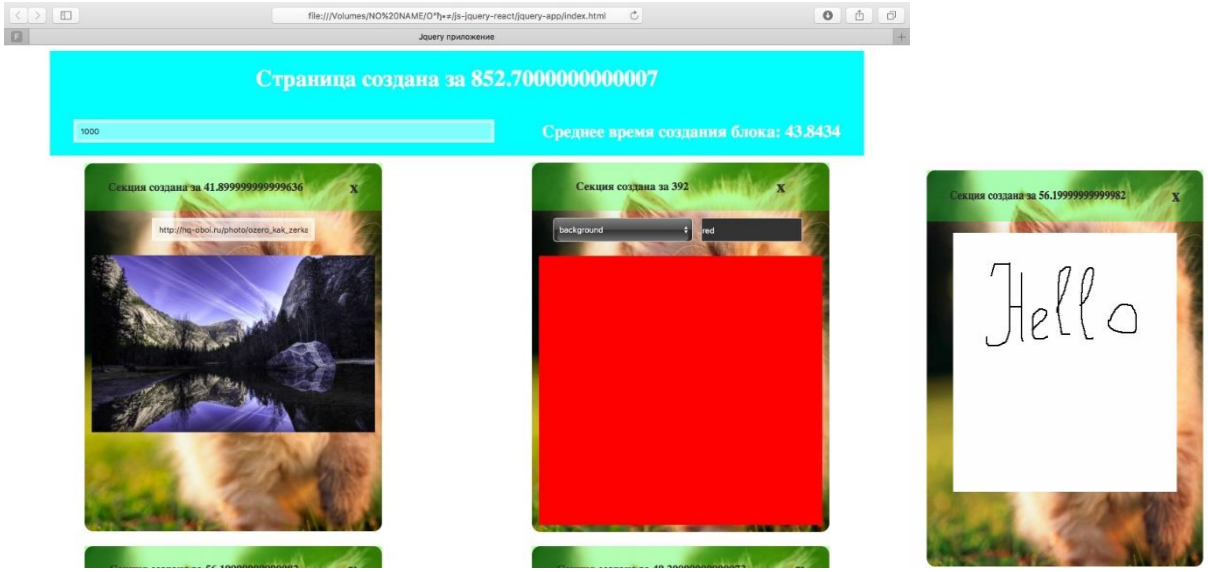

Рисунок 16 – Результат ToDo-приложения на jQuery при n=1000

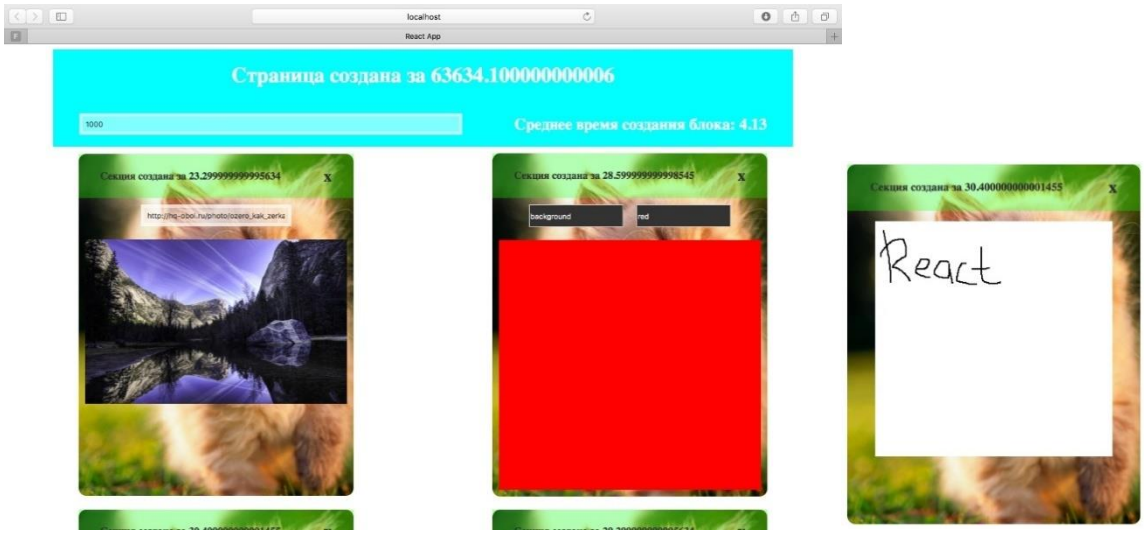

Рисунок 17 – Результат ToDo-приложения на React при n=1000

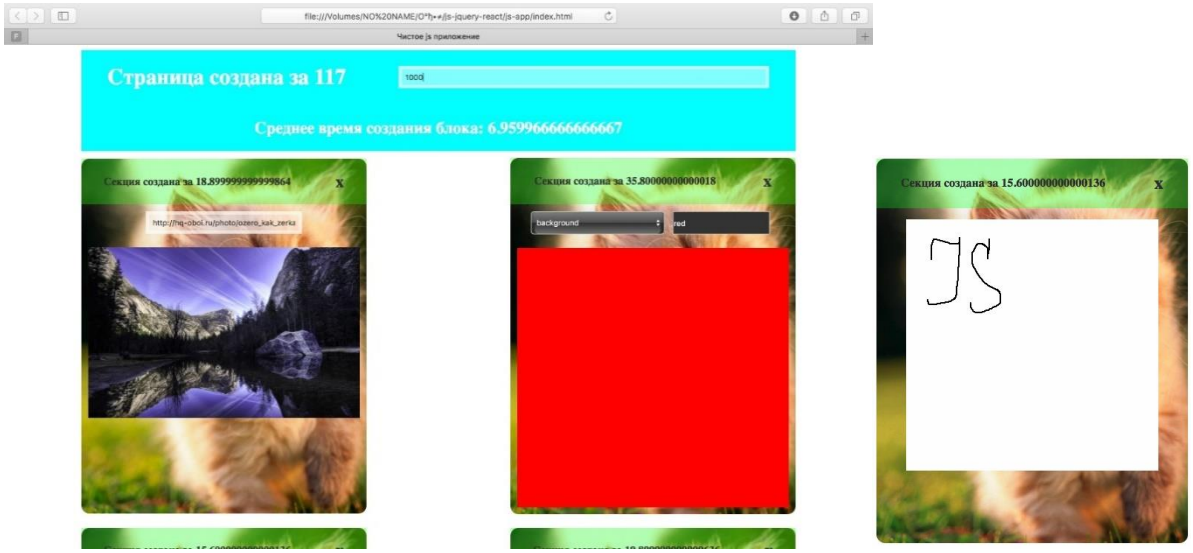

Рисунок 18 – Результат ToDo-приложения на JavaScript при n=1000

Таблица 3 – Результаты ToDo-приложения при n=1000

Показатель	jQuery	React.js	JavaScript
Время создания страницы	852,70	63634,1	117
Среднее время создания блока	43,83	4,13	6,95
Время создания блока «картинка»	41,89	23,29	18,89
Время создания блока «стиль»	392	28,59	36,80
Время создания блока «рисовалка»	51,19	30,40	15,60

4.8 Результаты сравнительного анализа

В ходе исследования были разработаны три ToDo-приложения с помощью трех разных инструментов, а именно jQuery, React.js и JavaScript и на их основе был проведен практический анализ и сравнение по критериям. Данные сравнения представлены в таблице 4.

Таблица 4 – СравнениеJQuery, React.js и JavaScript

Критерий	JQuery	React.js	JavaScript
Простота развертывания	Скачивание с сайта jQuery.com и подключение и подключение собственного скрипта: <scripttype="text/javascript" src="JQuery-3.3.1.min.js"></script> <script type="text/javascript" src="all.min.js"></script>	Использование Create React App и npm: npm install -g create-react-app npm start	Подключение отдельного файл со скриптом, следующим образом: <script type="text/javascript" src="all.min.js"></script>
Простота использования	Не нужно много писать, так как сценарии уже существуют	Не нужно много писать, так как сценарии уже существуют	Необходимо писать свои собственные сценарии, которые отнимают много времени
Совместимость	Кроссплатформен-ный	Кроссплатформен-ный	Необходимо обрабатывать совместимость с несколькими браузерами, написав собственный код.
Длина кода	268 строк	250 строк	753 строки
Скорость обработки	852,7	63634,1	117
Объем файлов	102 КБ	222,5 МБ	37 КБ

Исходя из вышеизложенной таблицы, можно сказать, что самой быстрой скоростью обработки обладает приложение написанное, с помощью JavaScript, но оно самое большое относительно длины кода. Несмотря на то, что приложение имеет самую быструю скорость обработки и занимает меньше места на диске,

данное приложение заняло наиболее большее время при разработке, нежели другие два.

Приложение, написанное с помощью React занимает наибольшее время для обработки и имеет наибольший объем файлов, однако время на разработку данного приложения ушло значительно меньше, чем на приложении с нативным JavaScript.

Приложение, написанное с помощью jQuery имеет средние показатели по вышесказанным критериям.

Кроссплатформенность данных приложений было проверено на нескольких браузерах, включая Safari.

Выводы по главе

В заключительной главе была проведена работа над практическим сравнением методологий управления объектной моделью веб-документа. Были разработаны три одинаковых ToDo-приложения с помощью трех разных методологий. на основе их работы произведен сравнительный анализ по выбранным критериям.

Заключение

В выпускной квалификационной работе были проанализированы отечественная и зарубежная литература. Рассмотрены и изучены методологии управления объектной моделью документа. На основе теоретического анализа выбраны критерии сравнения для практического анализа. В ходе практического анализа были созданы и рассмотрены три ToDo-приложения, которые разрабатывались с помощью jQuery, React.js и JavaScript.

На основе работы разработанных приложений был проведен сравнительный анализ методологий управления объектной моделью веб-документа по следующим критериям:

1.Простота развертывания.
2.Простота использования.
3.Совместимость.
4.Длина кода.
5.Скорость обработки кода.
6.Объем файлов.

Результаты работы показали, что нативный JavaScript является наиболее эффективным методом клиентской разработки, так как приложение, разработанное с помощью JavaScript имеет наибольшую скорость обработки и занимает наименьшее пространство на диске, поэтому есть причины его использования. Однако разработка заняла достаточно продолжительное время, так как требует глубоких знаний в языке, а также длина кода скрипта превышает длину кода других приложений в 2,5 раза. Но библиотеки jQuery и React.js, сокращают время разработки требуемого продукта, так как имеют готовые сценарии, что сокращает длину кода и время написания. Таким образом, лучше зависеть от JQuery или React.js для исходных версий продукта, которые требуют наискорейшего выхода на рынок.

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ВКР – Выпускная Квалификационная Работа

AJAX – Asynchronous Javascript And Xml

API – Application Programming Interface

CSS – Cascading Style Sheets

DOM – Document Object Model

HTML – HyperText Markup Language

MVVM – Model-View-ViewModel

XHTML – Extensible Hypertext Markup Language

SPA – Single Page Application

URL – Uniform Resource Locator

W3C – World Wide Web Consortium

WHATWG – Web Hypertext Application Technology Working Group

СПИСОК ТЕРМИНОВ

Веб-приложение: это клиет-серверное приложение, в котором клиентом выступает браузер.

Фреймворк: программный продукт, который упрощает разработку благодаря содержанию базовых модулей.

ToDo-приложение: базовое приложение, показывающее основную функциональность библиотеки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. БИБЛИОТЕКА JQUERY, 2016 [Электронный ресурс]. – Режим доступа: https://tyapk.ru/storage/app/media/uploaded-files/08%20jQuery.pdf (дата доступа: 11.10.2017).
2. Введение в React. Что такое React., 2017 [Электронный ресурс]. – Режим доступа: https://metanit.com/web/react/1.1.php (дата доступа: 11.12.2017).
3. Выразительный JavaScript: Document Object Model (объектная модель документа), 2014 [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/243815/ (дата обращения: 3.11.2017).
4. Зудилова Т.В., Буркова М.Л., Web-программирование JavaScript, СПб: НИУ ИТМО, 2012. – 68 с.
5. Основы Vue.js. Что такое Vue.js., 2017 [Электронный ресурс]. – Режим доступа: https://metanit.com/web/vuejs/1.1.php (дата доступа: 10.12.2017).
6. Какой JavaScript Framework используете вы? Опрос среди JS-разработчиков JavaScript, 2015 [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/257825/ (дата доступа: 11.10.2017).
7. Карышев А.А., Афанасьев В.Р., Разработка web-приложения для автоматизированной генерации документов на основе docx-шаблонов // Известия ТулГУ. Технические науки. 2017. Вып.5. С.294-296
8. Лучшие JavaScript фреймворки, библиотеки и инструменты в 2017, 2017 [Электронный ресурс]. – Режим доступа: https://proglib.io/p/best-javascript-frameworks-2017/ (дата доступа: 10.11.2017).
9. Матвеев А.Г., Якубайлик О.Э., Разработка веб-приложения для обработки и представления пространственных метаданных геопортала // Сибирский журнал науки и технологий. 2012. С.49-54
10. Медведев Ю.С., Терехов В.В., Достижение максимальной производительности AJAX-приложений // Вестник АГУ. 2013

11. Миронов В.В., Гусаренко А.С., Концепция управления XML-данными на основе динамических DOM-объектов // Вестник УГАТУ. 2012. №3. С. 159-172.
12. Основы JavaScript, 2015 [Электронный ресурс]. – Режим доступа: https://html5book.ru/osnovy-javascript/ (дата доступа: 10.11.2017).
13. Работа с DOM. Введение в DOM., 2015 [Электронный ресурс]. – Режим доступа: https://metanit.com/web/javascript/8.1.php (дата доступа: 3.11.2017).
14. DOMAPI, 2017 [Электронный ресурс]. – Режим доступа: https://htmlhook.ru/dom-api.html (дата доступа: 7.10.2017).
15. DOM и JavaScript, 2017 [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/DOM/DOM_Reference/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5 (дата доступа: 7.11.2017).
16. DocumentObjectModel, 2017 [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Document_Object_Model (дата доступа: 3.10.2017).
17. Document Object Model (DOM) Level 1 Specification // World Wide Web Consortium. – 1998. – С.7-13. – Режимдоступа:https://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/DOM.pdf (датаобращения 3.10.2017).
18. Document Object Model (DOM) Level 2 HTML Specification // World Wide Web Consortium. – 2003. – С.11-15. – Режимдоступа:https://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109/DOM2-HTML.pdf (датаобращения: 3.10.2017).
19. Document Object Model (DOM) Level 3 Core Specification // World Wide Web Consortium. – 2004. – С.11-15. – Режимдоступа:https://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/DOM3-Core.pdf (датаобращения: 3.10.2017).
20. Chaffer J., Swedberg K., JQuery Reference Guide // Packt Publishing Ltd., 2010. – Режимдоступа:https://books.google.ru/books?hl=ru&lr=&id=JpEQJmidAbUC&o

i=fnd&pg=PT35&dq=jQuery&ots=XkqFnYkvj-
#v=onepage&q=jQuery&f=false(датаобращения: 7.11.2017).

21. Lennon J., Compare JavaScript frameworks // IBM. developerWorks. – 2010. – Режимдоступа:https://www.ibm.com/developerworks/library/wa-jsframeworks/index.html (датаобращения: 7.11.2017).
22. Manipulating Documents: The Document Object Model (DOM), 2003 [Электронныйресурс]. – Режимдоступа: http://www.informit.com/articles/article.aspx?p=31354 (датаобращения: 7.10.2017).
23. McLaughlin B., Part 5: Managing the DOM // IBM. developerWorks. – 2006. – Режимдоступа: https://www.ibm.com/developerworks/ru/library/wa-ajaxintro5/index.html (датаобращения: 8.11.2017).
24. Orchard L.M., Pehlivanian A., Koon S., Jones H., Professional JavaScript Frameworks: Prototype,YUI, ExtJS, Dojo and MooTools // Wrox Press Ltd. Birmingham, UK, 2009. – Режим доступа:https://dl.acm.org/citation.cfm?id=SERIES11879.1795768 (датадоступа: 13.11.2017).
25. Watts N., Learning jQuery by Ralph Steyer // ACM New York, NY, USA, 2014. – Режим доступа:https://dl.acm.org/citation.cfm?id=2557837 (дата доступа: 13.12.2017).
26. WorldWideWebConsortium, 2017 [Электронный ресурс]. – Режим доступа: https://www.w3.org/(дата доступа: 3.12.2017).