МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ П.М. МАШЕРОВА»

Факультет математики и информационных технологий

Кафедра прикладного и системного программирования

КУРСОВАЯ РАБОТА   
  
по дисциплине «Основы алгоритмизации и программирования»

РАЗРАБОТКА ПРИЛОЖЕНИЯ «TODO APPLICATION» НА ФРЕЙМВОРКЕ JAVASCRIPT- ANGULAR

Козлов Никита Юрьевич,

1 курс 14 группа

Руководитель:

Никитин Александр Игоревич

Старший преподаватель

Витебск, 2020

**Реферат**

Курсовая работа 18 с., 5 рис., 9 лист., 13 источников.

Объект исследования – фреймворк Angular 9.

Предмет исследования – возможности фреймворка Angular 9 для создания бытового приложения.

Цель работы – разработать приложение, представляющее собой заполняемый и изменяемый список дел, используя фреймворк языка JavaScript.

Методы исследования: анализ, синтез, изучение документации, практическая реализация.

**Содержание**

[Введение 4](#_Toc41940352)

[1 Front-end разработка 8](#_Toc41940353)

[1.1 Варианты front-end разработки 8](#_Toc41940354)

[1.2 Характеристика понятия front-end разработки 8](#_Toc41940355)

[1.3 Разнообразие front-end разработки 13](#_Toc41940356)

[2 Программная реализация приложения 16](#_Toc41940357)

[3 Интерфейс 20](#_Toc41940358)

[Заключение 23](#_Toc41940359)

[Список использованных источников 24](#_Toc41940360)

[Приложение А 25](#_Toc41940361)

# Введение

Сейчас существует огромное количество самых разных менеджеров задач на любой вкус. Однако у них есть недостаток: они требуют регистраций и загрузок, а так же расчитаны на долгосрочное пользование.

Сфера применения планировщиков задач огромна: составление планов работ, списка покупок, текущих дел и т.д. Без хорошего планировщика задач в наши дни трудно, а самое главное — он должен быть простым, чтобы ничего не отвлекало вас от дел. Очень удобно, работая за компьютером или ноутбуком, иметь при себе список задач. Чаще всего придется либо писать его от руки, либо скачивать приложение, что занимает время и место на вашем девайсе. Планировщик на Angular удобен, т.к. его можно просто открыть в браузере и при желании можно быстро что- либо поменять, или добавить новую задачу.

Он настолько прост, что даже не требует регистрации!

Для такого приложения не требуются google — аккаунты и прочее. Оно просто в применении и не требует времени, чтобы разбираться в нем. Это приложение для тех задач, которые нужно выполнить непосредственно во время рабочего сеанса. Поэтому оно очень помогает в планировке своих ближайших действий, чтобы ничего не забыть во время работы.

1. Изучить документацию и уроки по разработке приложений на Angular 9.
2. Вспомнить навыки разработки на HTML.
3. Разработать функциональные возможности и интерфейс приложения.
4. Реализовать программно выполняемые функции приложения.

# 

# 1 Front-end разработка

## 1.1 Варианты front-end разработки

Для front-end разработки сайтов и веб-приложений используется язык программирования JavaScript, а также его фреймворки.

Существуют различного рода фреймворки и библиотеки, самые популярные из которых: Angular, Node.js, React. Кроме обязательного JavaScript, в front-end разработке используются также и другие языки, хоть и не так часто. Также для front-end разработки сайтов используют CMS. Самый популярный выбор — WordPress.

Технологии, которые используются для front-end разработки:

* HTML и CSS (в том числе сетки и CSS-фреймворки, спецификации W3C и WHATWG, HTML5/CSS3 POLYFILLS)
* Препроцессоры CSS (SASS, LESS, STYLUS и т. д.)
* JavaScript
* Популярные фреймворки и библиотеки: jQuery, Angular, React.JS, Backbone.js и т. д.)
* OOCSS / BEM / SMACSS
* SVG
* DOM
* HTML5 API
* Популярные CMS (WordPress, Drupal, Joomla и т.д.)

## 1.2 Понятие front-end разработки

Front-end (фронт-энд) –  [клиентская](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) сторона [пользовательского интерфейса](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), т.е. все, что видит пользователь, открывая веб-страницу, и с чем он взаимодействует.

**HTML**

**HTML** (от англ. HyperText Markup Language – «язык гипертекстовой разметки») - стандартизированный язык разметки документов во [Всемирной паутине](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0). Большинство [веб-страниц](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) содержат описание разметки на языке HTML (или [XHTML](https://ru.wikipedia.org/wiki/XHTML)). Язык HTML интерпретируется [браузерами](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80); полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

HTML (HyperText Markup Language) говорит браузеру, каково содержание страницы, например, «заголовок», «параграф», «список», «элемент списка».

Тег - элемент языка разметки гипертекста. Более правильное название - дескриптор. Используют только два тега - открывающий, или начальный, и закрывающий, или конечный, или еще дополнительно в зависимости от реализации языка возможно применение одиночного тега и тега пустого элемента. Например, тег отступа абзаца может оформлять абзац между <p> </p>, а может в одиночной форме до следующего первого попавшегося <p>.

Пример пустого элемента: <HR></HR> - разрыв текста без сохранения отступов, но вы можете использовать специальную форму записи тега, используя меньшее число символов: <HR/>

Каждый тег состоит из имени, за которым может следовать список необязательных атрибутов, все они находятся внутри угловых скобок < >. Содержимое скобок никогда не выводится в окне браузера. Имя тега, как правило, представляет собой аббревиатуру его функции, что облегчает его запоминание. Атрибуты являются свойствами, которые расширяют или уточняют функцию тега. Как правило, имя и атрибуты внутри тега не чувствительны к регистру. Однако значения определенных атрибутов могут быть чувствительны к регистру. Это относится, в частности, к именам файлов и URL.

Документ должен начинаться тегом <HTML> и заканчиваться закрывающим тегом </HTML>. Между ними должны находится теги <HEAD> открывающий и </HEAD> закрывающий с названием и заголовком, а также <BODY> открывающий и </BODY> закрывающий с текстом страницы. Название страницы на латинском языке ограничивается тегами <TITLE> открывающий и </TITLE> закрывающий. Конечный тег имеет то же имя, что и начальный, но перед ним стоит слеш (/). Его можно рассматривать как "выключатель" тега. Конечный тег никогда не содержит атрибутов.

Браузеры прекрасно понимают html и могут интерпретировать его в понятном виде. Вообще любая страница сайта – это html-код, который браузер переводит в дружественный вид для пользователя.

Некоторые теги не имеют завершающих тегов, потому что они используются для размещения отдельных (автономных) элементов на странице. Одним из них является тег изображения <img>. Он просто помещает графику в поток страницы. Другие автономные теги - это разрыв строки (<br>), горизонтальная линия (<hr>) и теги, содержащие информацию о документе и не влияющие на содержимое, выводимое на экран, такие как <meta> и <base>.

Файлы, созданные на html, имеют расширения .html или .htm. Чтобы создать файл html, достаточно открыть блокнот и сохранить его с расширением .html. Однако большинство Web-мастеров используют для своей работы и редакторы, и написание текстов непосредственно на HTML, чаще всего в Visual Studio Code или Sublime Text .

На листинге 1.2.1 приведен пример со структурой типичной html страницы. Обратите внимание, что html код делится на две области: заголовок и тело страницы.

Листинг 1.2.1 – Пример со структурой типичной html страницы

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Название страницы</title>

Заголовочные теги (подключение стилей, скриптов)

...

</head>

<body>

<div class="header">

Шапка сайта

</div>

...

<div class="content">

Основная часть сайта

</div>

...

<div class="footer">

Низ страницы

</div>

...

</body>

</html>

**CSS**

Если HTML представляет собой скелет (структуру) сайта, т.е. необходим для структурирования содержания страницы, то CSS – это язык таблицы стилей, которые отвечают за то, как это будет выглядеть (какой будет шрифт: размер, цвет, толщина; как будут располагаться элементы на сайте относительно друг друга и т.п.), т.е. необходим для того, чтобы форматировать это структурированное содержание. Поэтому HTML и CSS довольно тесно связаны между собой.

CSS (**Cascading Style Sheets – каскадные таблицы стилей**) – это фактически язык стилей, который определяет отображение HTML-документов. CSS работает со шрифтами, с цветами символов и фона, с полями, со строками, с высотой и с шириной элементов отображения, с фоновыми изображениями, с позиционированием элементов и со многим другим.

Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или [методах](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4) вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими [шрифт Брайля](https://ru.wikipedia.org/wiki/%D0%A8%D1%80%D0%B8%D1%84%D1%82_%D0%91%D1%80%D0%B0%D0%B9%D0%BB%D1%8F).

Использование CSS облегчает создание качественных сайтов, позволяя задать стили отдельных элементов страниц сайта в особых css-файлах, чтобы в дальнейшем быть уверенным в том, что все страницы сайта будут выдержаны в едином стиле.

Для того, чтобы подключить CSS к HTML используют команду

<link rel="stylesheet" href="style.css">

Эта команда используется, если CSS файл находится отдельно от HTML файла, но также можно описывать CSS и в самом HTML используя тег <style></style> который должен находиться в теге <head></head>

**Sass**

**Sass** (Syntactically Awesome Stylesheets) — модуль, включенный в Haml. Sass — это метаязык на основе CSS, предназначенный для увеличения уровня абстракции CSS кода и упрощения файлов каскадных таблиц стилей.

Язык Sass имеет два синтаксиса:

* **sass** — отличается отсутствием фигурных скобок, в нём вложенные элементы реализованы с помощью отступов;
* **SCSS** (**Sassy CSS**) — использует фигурные скобки, как и сам CSS.

**JavaScript**

JavaScript **–** [мультипарадигменный](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Поддерживает объ[ектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [императивный](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [функциональный](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) стили.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам [приложений](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0). Наиболее широкое применение находит в [браузерах](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) как язык сценариев для придания [интерактивности](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [веб-страницам](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0).

Основная идея JavaScript состоит в возможности изменения значений атрибутов HTML-контейнеров и свойств среды отображения в процессе просмотра HTML-страницы пользователем. При этом перезагрузки страницы не происходит. Условно говоря, **JS помогает сделать HTML-разметку более интерактивной**, т.е. с помощью него реализуется практически вся динамика, которую вы можете наблюдать на сайте.

Таким образом, мы можем задавать изменение любых визуальных параметров для любых элементов по выполнении любых событий (коих превеликое множество).

Данный язык программирования считается одним из самых легких к изучению, поэтому он так широко применяется front-end-разработчиками сайтов, которые далеко не всегда являются программистами.

Можно сделать вывод, что JavaScript является незаменимым инструментом для того, чтобы создавать анимационные эффекты на сайте. При этом, переходя к изучению JS, необходимо хорошо разобраться в HTML и CSS, так как большинство программного кода будет «взывать» к их синтаксису.

Для добавления JavaScript-кода на страницу можно использовать теги <script></script>, которые рекомендуется, но не обязательно, помещать внутри контейнера <head>. Контейнеров <script> в одном документе может быть сколько угодно. Атрибут «type='text/javascript'» указывать необязательно, данное значение используется по умолчанию.

<script type="application/javascript"> </script>

Если же файл с JavaScript находится отдельно, то подключить его к HTML можно с помощью команды.

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

## 1.3 Разнообразие front-end разработки

Как правило, одной back-end - разработки достаточно для полноценного существования проекта. Даже от верстки можно уйти, если использовать фреймворки наподобие Bootstrap. Более того, большинство существующих сайтов — это как раз дизайн, верстка и back-end. Но есть ситуации, когда такая схема не работает.

Front-end - часть сайта или приложения — это то, что вы видите у себя в браузере, эта часть активно взаимодействует с серверной (back-end) частью, которая находится на каком-либо сервере, постоянно обмениваясь с ней данными.  
С технической точки зрения фронтенд-часть приложения — это набор файлов, среди которых есть файлы HTML, CSS и JavaScript, картинки и т.п. Работу с CSS и HTML относят к вёрстке, JavaScript — к программированию. Оба этих направления предлагают большое количество инструментов и технологий для работы, активно развиваются и требуют большого количества знаний. Особенно это относится к JavaScript, на котором написано гигантское количество фреймворков и библиотек для «всё более эффективного» создания веб-приложений.  
Программист с большим опытом работы с React сможет быстро написать приложение на React, наличие опыта работы с Angular даёт скорость работы с Angular и так далее. Всё достаточно очевидно. Сами по себе фреймворки экономят время разработки тем, что дают решения типичных проблем и задач. Эти решения по сути могут быть очень близки друг другу, и разница между ними может заключаться в синтаксисе или парадигме программирования.  
Скорость разработки с использованием определённого фреймворка зависит от опыта и квалификации программистов, которые на нём пишут, сам фреймворк здесь имеет второстепенное значение.

Практически любой современный популярный фреймворк достаточно хорош, чтобы на нём создать практически любое приложение, которое может потребоваться современному бизнесу.

Самыми популярными фреймворками для фронтенд-разработки можно назвать Vue.js React и Angular. Каждый из них предназначен для решения своего спектра задач и имеет различную степень сложности: Vue.js - самый легкий (но и с наименьшим сообществом), React - средней сложности, Angular - высокой сложности.

Библиотека — это тоже JavaScript код, уже написанный другими разработчиками. Он предоставляет удобный набор решений, чтобы выполнять заготовленные действия, которые регулярно требуются в работе. В веб-программировании есть большое разнообразие таких библиотек, которые упрощают написание отзывчивых интерфейсов для веб-сайтов. В их числе самые известные — это React, Vue и Angular.  
Они используются немного в разных сферах. Angular — один из самых старых. Он чаще используется в корпоративном сегменте, на больших проектах, с поддержкой легаси кода, который был написан несколько лет назад. React используется повсеместно. Он первый в списке, когда пишут новые интерфейсы и веб-сервисы. Про Vue можно сказать, что он простой и новый.

Особо следует заметить, что технологии в части front-end разработки развиваются просто **сумасшедшими темпами**! Мода на то или иное решение или инструмент меняется с угрожающей стабильности скоростью.

# 2 Программная реализация приложения

Для реализации front-end части приложения использовались язык разметки HTML, формальный язык описания внешнего вида документа – CSS, а фреймворк JS- Angular.

При создании данного списка задач использовалось следующее:

Листинг 2.1 – Код для добавления кнопки, при нажатии на которую текст зачеркивается и помечается как “выполненный”

  onToggle(id:number) {  
 const idx = this.todos.findIndex(t=>t.id===id)  
 this.todos[idx].completed =!this.todos[idx].completed  
}

<span [class.done]="***todo***.completed">  
 <input type="checkbox" [checked]="***todo***.completed" (change)="onChange(***todo***.id)">

Листинг 2.2 – Код для создания кнопки “Удалить”

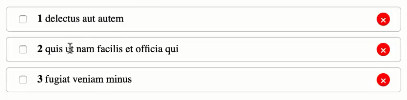
<button class="rm" (click)="removeTodo(***todo***.id)">&times;</button>



**Рисунок 2.1 – кнопкa “Add todo ”**

Листинг 2.3 – Код для удаление выполненной задачи

removeTodo(id: number) {  
 this.todos = this.todos.filter(t=>t.id!==id)  
}



**Рисунок 2.2 – кнопкa “Удалить”**

Листинг 2.4 – Код, отвечающий за показ всех задач и элементов

<ul *\*ngIf*="todosService.todos.length; else ***noTodos***">  
 <li *\*ngFor*="let ***todo*** of todosService.todos; let ***i*** = index">  
 <span [class.done]="***todo***.completed">  
 <input type="checkbox" [checked]="***todo***.completed" (change)="onChange(***todo***.id)">  
 <strong>{{***i***+1}}</strong>  
 {{ ***todo***.title}}  
 </span>  
  
 <small>{{***todo***.date| date:'short'}}</small>  
  
 <button class="rm" (click)="removeTodo(***todo***.id)">&times;</button>  
 </li>  
</ul>  
  
<ng-template #***noTodos***>  
 <p style="text-align: center;">No todos</p>  
</ng-template>

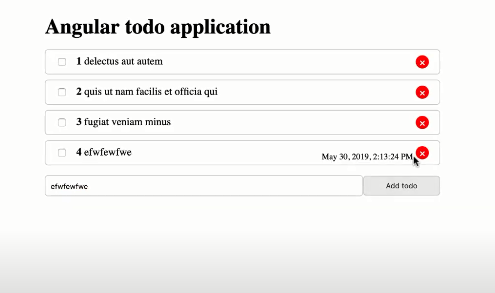
# 3 Интерфейс

В разделе представлены результаты выполнения основных функциональных частей front-end разработки «TODO LIST». Описание результатов реализации приводится вместе с описанием интерфейса приложения.



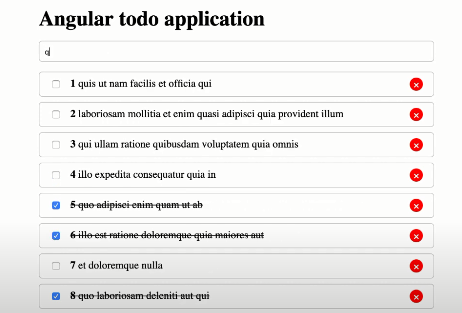
**Рисунок 3.1 – Окно списка дел “TODO LIST”**

На рисунке 3.1 показан внешний вид сайта. При нажатии на строку для ввода текста можно начать вводить свою первую задачу на день. После того, как задача написана, то в список задач ее можно добавить путем нажатия левой кнопкой мыши на “+” или же нажатием на клавиатуре кнопку “Enter”.



**Рисунок 3.2 – Добавленные задачи**

На рисунке 3.2 показаны добавленные задачи в список дел.



|  |  |
| --- | --- |
|  | **Рисунок 3.3 – Демонстрация как выделяются выполненные задачи** |

На рисунке 3.3 показано как выделяются выполненные задачи.

# Заключение

В процессе выполнения курсовой работы были изучены основные возможности языков HTML, CSS и фреймворкаJS, освоены и закреплены основные принципы работы с языков программирования JavaScript, а также были расширены познания в области Front-end разработки.

Были применены знания, полученные в процессе изучения курсов «Основы алгоритмизации и программирования». При выполнении курсовой работы была изучена и проработана литература по выбранной теме.

Так как “TODO LIST” создавался для пользователей, не имеющих специальных знаний в области программирования, требуется, чтобы оно было достаточно простым в понимании и использовании. Навигация по приложению должна быть интуитивно понятна.

Выполнение курсовой работы помогло закрепить материал и навыки, полученные на лекциях и лабораторных работах, систематизировать знание языков и научиться применять их к конкретным задачам.

Считаю, что цель, поставленная во введении, была достигнута: разработана программа, представляющая собой Web-приложение «TODO LIST» для создания списка дел.

В процессе выполнения представленной курсовой работы были решены следующие задачи:

- получены навыки разработки программного обеспечения на фреймворке JS- Angular;

- реализована возможность добавлять задачи, помечать как “выполненные” и удалять задачи.

Полученный результат соответствует поставленным задачам и цели.

Можно сделать вывод о том, что список дел помогает людям строить свой день более продуманно и эффективно.

# Список использованных источников

1. Крокфорд, Д. JavaScript: сильные стороны / Д. Крокфорд. — СПб: Питер, 2012 — 176 с.

2. Флэнаган, Д. JavaScript: Подробное руководство / Д. Флэнаган. — Символ-Плюс, 2008 — 1080 с.

3. Стефанов, С. JavaScript: Шаблоны / С. Стефанов — Символ-Плюс, 2016 — 272 с.

4. Хавербеке, М. Выразительный JavaScript. Современное веб-программирование / М. Хавербеке — СПб: Питер, 2019 — 480 с.

5. Моррисон, М. Изучаем JavaScript / М. Моррисон — СПб: Питер, 2012. — 608 с.

6. Роббинс, Дж. HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Дж. Роббинс; [пер. с англ. М. А. Райтман]. — 4-е издание. — М.: Эксмо, 2014. — 528 с.

7. Простыми словами о «фронтенде» и «бэкенде»: что это такое и как они взаимодействуют [Электрон. ресурс]. – 2012. <https://tproger.ru/translations/frontend-backend-interaction/>.

8. Современный учебник JavaScript [Электрон. ресурс]. – 2012. <https://learn.javascript.ru/>.

9. Фримен, Э., Робсон Э. Изучаем программирование на JavaScript / Э. Фримен, Э. Робсон — СПб: Питер, 2015. — 640 с.

10.Крокфорд, Д. Как устроен JavaScript / Д. Крокфорд. — СПб: Питер, 2019 — 304 с.

11. Минник, К. Холланд Е. JavaScript для чайников / К. Минник, Е. Холланд — Диалектика, 2019 — 320 с.

12. Поллок, Дж. JavaScript: Руководство разработчика / Дж. Поллок — СПб:Питер, 2011 — 544 с.

13. Флэнаган, Д. JavaScript: Карманный справочник / Д. Флэнаган — Вильямс, 2019 — 320 с.