**МИНИСТЕРСТВО ОБРАЗОВАНИЯ АЗЕРБАЙДЖАНСКОЙ РЕСПУБЛИКИ**

**ЗАКРЫТОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО**

**«АЗЕРБАЙДЖАН ХАВА ЙОЛЛАРЫ»**

**НАЦИОНАЛЬНАЯ АКАДЕМИЯ АВИАЦИИ**

****

**«Допущен(а) к защите»**

Заведующий кафедрой, д.т.н., профессор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Меликов А.З**,

«\_\_\_» «\_\_\_\_\_\_\_\_\_» 2021

**ВЫПУСКНАЯ РАБОТА**

**Факультет:** Физико-технологический

**Кафедра:** Информационные технологии

**Специальность:** 050631 –Компьютерная инженерия

**Специализация**: Компьютерная инженерия

**Тема выпускной**

**работы: “Создание обучающей программы силовой установки вертолета (двигатель ТВ3-117ВМ)”**

**Руководитель**

**выпускной работы:\_\_\_\_\_\_к.ф.-м.н., доц. Ахмедов Лютвияр Нурмамед оглы**

**Студент(ка): \_\_\_\_\_\_\_\_ Салманов Эльвин Вали оглы**

**Группа:** 1457r

**Нормоконтроль:** \_\_\_\_\_\_\_\_\_\_ **Аманов Рашад Шахлар оглы**

(подпись)

**Рецензент: \_\_\_\_\_\_\_\_\_**

(подпись)

**БАКУ – 2021**

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ АЗЕРБАЙДЖАНСКОЙ РЕСПУБЛИКИ**

**ЗАКРЫТОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО**

**«АЗЕРБАЙДЖАН ХАВА ЙОЛЛАРЫ»**

**НАЦИОНАЛЬНАЯ АКАДЕМИЯ АВИАЦИИ**

**Факультет**: Физико-технологический

**Kaфедра**: Информационные технологии

**Специальность**: 050631 –Компьютерная инженерия

**Специализация:** Компьютерная инженерия

**Студент: Салманов Эльвин Вали оглы**

**Группа:** 1457r

**ЗАДАНИЕ НА ВЫПУСКНУЮ РАБОТУ**

**1.Тема выпускной работы: “Создание обучающей программы силовой установки вертолета (двигатель ТВ3-117ВМ)” (утверждена приказом № от \_\_\_\_\_\_2021)**

**2. Первичные данные по выпускной работе:\_\_\_Сведения о силовой установке вертолета (двигатель ТВ3-117ВМ), сведения о принципах построения обучающих программ и типах обучающих программ,**

**сведения об элементах управления в сценариях обучающих программ. Информация о средствах и технологиях, используемых при разработке обучающих систем (HMTL, CSS, Bootstrap ,JavaScript, React js) ------------------------------------------------------------------------------------**

**3. План работы: Введение; Глава 1.** **Принципы построения обучающей программы; Глава 2. Средства, используемые при создание обучающей программы; Глава 3. Разработка обучающей программы силовой установки вертолета (двигатель ТВ3-117ВМ); Заключение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**4. Количество и содержание графических материалов – 1.Классификация структурного построения. 2. Структурная схема тестирующей системы обучения. 3. Интерфейс программного модуля**

**5. Рекомендации к выполнению разделов выпускной работы:** изучение классификации и типов обучающих программ; анализировать функциональные возможности элементов управления в сценариях обучающих программ; изучение возможностей программных средств и технологий, используемых при разработке обучающих программ; описание интерфейса, функциональных возможностей и информационной базы данных разрабатываемой системы.

**Задание получено (дата) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата сдачи выпускной работы : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Руководитель выпускной работы:**  **Ахмедов Лютвияр Нурмамед оглы**

**(**подпись) (ФИО руководителя)

**Заведующий кафедрой: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Меликов Агаси Зарбали оглы**

**(** подпись) (ФИО)

**Задание принял(а): \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Салманов Эльвин Вали оглы**

(подпись студента) (ФИО студента)

**ЭТАПЫ ВЫПОЛНЕНИЯ РАБОТЫ**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Наименование этапа работы** | **Сроки** | **Замечания**  **Руководителя** |
| **1** | **Сбор и обработка первичных данных** | 01.10 - 01.12.2020 | **выполнено** |
| **2** | **Изучение возможностей технологий, программных средств, используемых при разработке обучающей программы** | 01.04.2018 - 30. 04.2021 | **выполнено** |
| **3** | **Создание обучающей программы** | 01.12.2020 - 30.03.2021 | **выполнено** |
| **4** | **Оформление выпускной работы и подготовка к предварительной защите** | 01.05 – 01.06.2021 | **выполнено** |

**РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА**

1. <http://www.ito.su/2001/ito/II/II-4-7.html>

Технология разработки и использования электронных учебников. Калинин Илья Александрович (МГПУ г.Москва)

2. *Стивен Шафер.* HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2010.

3.<https://www.w3schools.com/>

4. <https://ru.reactjs.org/docs/getting-started.html>

5. Jake Spurlock. Bootstrap. Responsive Web-Development. — O'Reilly, 2013.

**АННОТАЦИЯ**

Целью данной выпускной работы является разработка обучающей программы силовой установки вертолета (двигатель ТВ3-117ВМ), содержащий информацию о непосредственно самом двигателе, которой будут пользоваться кафедры «Летная эксплуатация воздушных судов» и «Летательные аппараты и авиационные двигатели», в целях обучения и подготовки студентов-пилотов.

Обоснована актуальность темы, сформулирована цель и вытекающие из нее задачи. Осуществлен анализ существующих обучающих программ и проведен сравнительный анализ. Рассмотрены возможности HTML, CSS, JAVASCRIPT, библиотеки Bootstrap и библиотеки JavaScript - React js. Описана структура веб-приложения и его работа.

**XÜLASƏ**

Bu buraxılış işinin məqsədi "hava gəmilərinin uçuş istismarı" və "uçuş aparatları və aviasiya mühərrikləri" kafedralarından istifadə edəcək bilavasitə mühərrik haqqında məlumatları özündə əks etdirən helikopterin güc qurğusunun təlim proqramının (TV3-117VM mühərriki) işlənib hazırlanmasıdır.

Mövzunun aktuallığı əsaslandırılmış, məqsəd və ondan irəli gələn vəzifələr formalaşdırılmışdır. Mövcud tədris proqramlarının təhlili aparılmış və müqayisəli təhlil aparılmışdır. HTML, CSS, JAVASCRİPT, Bootstrap kitabxanaları və JavaScript - React js kitabxanalarının imkanları nəzərdən keçirilir. Veb tətbiqin strukturu və onun işi təsvir edilmişdir.

**ОГЛАВЛЕНИЕ**

[**ВВЕДЕНИЕ** 7](#_Toc71801691)

[**ГЛАВА 1. ПРИНЦИПЫ ПОСТРОЕНИЯ ОБУЧАЮЩЕЙ ПРОГРАММЫ** 8](#_Toc71801692)

[**1.1. Классификация структурного построения** 8](#_Toc71801693)

[**1.2. Типы обучающих программ** 12](#_Toc71801694)

[**1.3. Элементы управления в сценариях обучающих программ** 14](#_Toc71801695)

[**ГЛАВА 2. СРЕДСТВА, ИСПОЛЬЗУЕМЫЕ ПРИ СОЗДАНИЕ ОБУЧАЮЩЕЙ ПРОГРАММЫ** 16](#_Toc71801696)

[**2.1. Язык разметки гипертекста HTML и каскадные таблицы стилей CSS** 16](#_Toc71801697)

[**2.1.1. Общие сведения о HTML** 16](#_Toc71801698)

[**2.1.2. Что из себя представляет CSS** 23](#_Toc71801699)

[**2.2. Язык JavaScript** 25](#_Toc71801700)

[**2.3. Bootstrap** 28](#_Toc71801701)

[**2.3.1. Из чего состоит Bootstrap** 29](#_Toc71801702)

[**2.3.2. Преимущества и недостатки Bootstrap** 29](#_Toc71801703)

[**2.3.3. Какую версию Bootstrap выбрать** 31](#_Toc71801704)

[**2.4. React js** 31](#_Toc71801705)

[**2.4.2. Ключевые особенности  React: декларативность, универсальность, компонентный подход, виртуальный DOM, JSX** 33](#_Toc71801706)

[**2.4.3. Рендеринг элемента в DOM** 34](#_Toc71801707)

[**ГЛАВА 3. Разработка обучающей программы силовой установки вертолета (двигатель ТВ3-117ВМ)** 35](#_Toc71801708)

[**3.1. Постановка задачи** 35](#_Toc71801709)

[**3.2. Структура главной страницы** 35](#_Toc71801710)

[**3.3. Создание главной страницы сайта** 36](#_Toc71801711)

[**3.4.Каркас(Шаблон) для Страниц** 42](#_Toc71801712)

[**3.5. Просмотр информации о двигателе** 43](#_Toc71801713)

[**3.6. Страница с информацией о Национальной Академии Авиации** 45](#_Toc71801714)

[**3.7. Страница с контактными данными НАА** 47](#_Toc71801715)

[**ЗАКЛЮЧЕНИЕ** 49](#_Toc71801716)

[**ЛИТЕРАТУРА** 50](#_Toc71801717)

# **ВВЕДЕНИЕ**

В настоящем постиндустриальном обществе роль информационных технологий чрезвычайно важна, они занимают сегодня центральное место в процессе интеллектуализации общества, развития его системы образования и культуры. Их широкое использование в самых различных сферах деятельности человека диктует целесообразность наискорейшего ознакомления с ними, начиная с ранних этапов обучения и познания.

Применение мультимедиа в сфере образования ряда развитых западных стран уже идет достаточно успешно и имеет следующие направления: видеоэнциклопедии; интерактивные путеводители; тренажеры; ситуационно-ролевые игры; электронные лектории; персональные интеллектуальные гиды по различным научным дисциплинам, являющиеся обучающими системами с использованием искусственного интеллекта; исследовательское обучение при моделировании изучаемого процесса в аналоговой или абстрактной форме; системы самотестирования знаний обучающегося; моделирование ситуации до уровня полного погружения — виртуальная реальность (для изучения языка — моделирование деловых переговоров на иностранном языке, моделирование положения на бирже при изучения экономических вопросов и т.д.);

Конечно обучающая программа не может заменить человека-преподавателя, но оно не только может дополнить и усовершенствовать деятельность преподавателя, а в некоторых областях, в которых развиваются самостоятельность, творческое мышление, оно сыграет уникальную роль, которую мы сейчас не можем еще осознать в полной мере.

Подводя итоги вышеизложенного можно отметить: проблема создания обучающих программ или курсов является актуальной. Объектом нашего исследования будет обучающая программа, а предметом – методика разработки сценария обучающей программы.

# **ГЛАВА 1. ПРИНЦИПЫ ПОСТРОЕНИЯ ОБУЧАЮЩЕЙ ПРОГРАММЫ**

## **1.1. Классификация структурного построения**

В настоящее время в процесс обучения активно внедряются программные технологии на базе персональных ЭВМ, применяемые для передачи ученику учебного материала и контроля степени его усвоения. При этом на рынке программного продукта за последнее десятилетие появилось достаточно большое количество обучающих систем, в том числе и автоматизированных (АОС), которые охватывают различные предметные области, и призваны решать задачи обучения на различных этапах жизни человека - от начальных классов средней школы до процесса обучения в высших учебных заведениях. Это приводит к тому, что ряд программ имеют крайне ограниченные функциональные возможности и не позволяют в полной мере усвоить обучаемым преподносимый материал, а в некоторых случаях вообще дискредитируют саму идею АОС на базе персональных ЭВМ.

В данной главе мы попытаемся обобщить принципы построения существующих АОС, а также классифицировать принципы их построения.

Известно, что любая программа представляет собой набор алгоритмов (компонентов), которые взаимодействия между собой решают поставленную задачу. При этом программа будет являться программной системой, если она представляет собой совокупность взаимосвязанных компонентов, каждый из которых выполняет вполне определенные функции. В общем случае любая обучающая программа может считаться программной системой, так как в ней обязательно присутствует компонента интерфейса пользователя, и компонента, реализующая предлагаемую методику. Автоматизированной обучающей системой будет являться любая АОС, так как согласно, ряд задач, например, отображение информации или анализ правильного ответа, выполняются без участия человека. Каждая АОС имеет определенную структуру на основе группы элементов с указанием связей между ними и дающее представление о системе в целом. Поэтому структура системы может быть охарактеризована по имеющимся в ней типам связей.

На основе проведенного анализа можно сделать вывод, что каждая обучающая система имеет четко выраженную структуру, и эти структуры можно классифицировать следующим образом:

Автоматизированные обучающие системы

Без обратной связи

(разомкнутые)

Презентационные

Тестирующие

С обратной связью

(замкнутые)

Имитационные

Тестирующие -обучающие

Без игровых элементов

С игровыми элементами

***Рис 1.1. Классификация структурного построения***

По структурным признакам взаимодействия обучающей системы с пользователем АОС подразделяются на два базовых класса (рис.1.1): разомкнутые (без обратной связи) и замкнутые (с обратной связью) системы, которые отличаются принципиальным подходом к процессу обучения.

В разомкнутых АОС не учитываются отклики учащихся на поставленные вопросы и не корректируется последовательность предъявления учебного материала в функции степени усвоения учащимся изучаемой темы. Здесь лишь выполняется определенная заранее, заданная программным путем, последовательность изложения урока или контрольных вопросов. При этом наиболее простыми из числа разомкнутых АОС являются системы с презентационной структурой, представляющей собой последовательное включение звеньев "АОС" и "Учащийся" (рис 1.2)

АОС

Учащийся

***Рис 1. 2. Структурная схема презентационной системы обучения***

В АОС данного типа присутствует только прямая информационная связь между системой и обучаемым, которому последовательно предоставляется визуальная информация с монитора ЭВМ. При этом обучаемый находится в режиме пассивного наблюдателя, от которого не требуется ни каких откликов по взаимодействию с АОС. Примером презентационной АОС может служить обучающая программа по курсу VisualBasic, разработанная автором, представляющая набор слайдов с демонстрационными примерами.

В тестирующих АОС без обратной связи (рис.1.3.) основной упор делается на выявление уровня знаний в определенный период учебного процесса. Используя различную методику, такие системы предъявляют обучаемому открытый или закрытый вариант вопроса (вопрос с вариантами выбора ответа). От учащегося ожидается отклик в виде ответа (управляющего воздействия) на поставленный вопрос. Ответ фиксируется в блоке фиксатора ошибок. По результатам опроса выставляется определенный балл, который служит критерием для результирующей оценки по степени усвоения учащимся требуемого учебного материала.

Информационное воздействие

**Учащийся**

**АОС**

**Фиксатор** **ошибок**

Управление воздействием

***Рис 1.3. Структурная схема тестирующей системы обучения***

Примером разомкнутой тестирующей АОС является программная система компании DrakePrometric, которая ориентированна на простое предъявление вопросов, для выявления знаний претендентов на тот или иной сертификат той или иной фирмы.

Наиболее широкими функциональными возможностями и высокой эффективностью в учебном процессе обладают АОС, где организована обратная связь между учащимся и обучающей системой.

В зависимости от характера воздействия со стороны АОС учащийся принимает определенное, достоверное с его точки зрения, решение, доказывающее, по его мнению, факт усвоения им поданного материала, и генерирует его на вход ЭВМ.

Весьма распространенным типом АОС среди замкнутых систем являются имитационные автоматизированные обучающие системы. Здесь функции ведущего "элемента" выполняет фактор моделирования реальной ситуации в той или иной сфере предметной области. Элемент обратной связью в виде реакции ученика на предъявляемый АОС учебный материал является основой непрерывного взаимодействия системы "АОС-обучаемый", так как то или иное воздействие на систему со стороны пользователя ведет сразу к ответной реакции со стороны обучающей системы. Примером подобных АОС могут служить всевозможные игровые тренажеры, имитаторы и т.п. В частности, имитационная программа компании Maxis позволяет имитировать развитие и управление городом. Здесь имитируются различные ситуации, начиная от размещения промышленных предприятий и транспортных сетей и заканчивая моделированием экстремальных ситуаций и путей их ликвидации.

В нотационных АОС используется комплексный подход в обучении. Программа не только обучает, но и одновременно проверяет полученные на текущий момент знания учащимся. Здесь важным фактором служит отклик учащегося на то или иное информационное воздействие. В зависимости от отклика, обучающая система может перестроить ход урока в том или ином направлении. В АОС данной структуры очень часто вводят игровые элементы. Как правило, такие АОС рассчитаны на учащихся младших возрастов, для которых игра является важным инструментом обучения. Здесь вопрос (информационное воздействие) ставится в виде игровой ситуации, на основании которой учащийся может находить верный или неверный выход (отклик). На основании принятого учащимся решения, АОС формирует следующий вариант игровой ситуацию. Примером такой системы может служить курс обучения английскому языку BridgetoEnglish.

## **1.2. Типы обучающих программ**

Рассмотренные принципы классификации структурного построения охватывают практически весь спектр существующих АОС, и позволяют автоматизировать процесс построения обучающих систем, путем разработки стандартных программных элементов, учитывая основные положения, рассмотренные в этой статье.

Основанием для классификации служат обычно особенности учебной деятельности обучаемых при работе с программами. Многие авторы выделяют четыре типа обучающих программ:

• тренировочные и контролирующие;

• наставнические;

• имитационные и моделирующие;

• развивающие игры.

Программы 1-го типа (тренировочные) предназначены для закрепления умений и навыков. Предполагается, что теоретический материал уже изучен. Эти программы в случайной последовательности предлагают учащемуся вопросы и задачи и подсчитывают количество правильно и неправильно решенных задач (в случае правильного ответа может выдаваться поощряющая ученика реплика). При неправильном ответе ученик может получить помощь в виде подсказки.

Программы 2-го типа (наставнические) предлагают ученикам теоретический материал для изучения. Задачи и вопросы служат в этих программах для организации человеко-машинного диалога, для управления ходом обучения. Так если ответы, задаваемые учеником, неверны, программа может “откатиться назад” для повторного изучения теоретического материала.

Программы 3-го типа (моделирующие) основаны на графически-иллюстративных возможностях компьютера, с одной стороны, и вычислительных, с другой, и позволяют осуществлять компьютерный эксперимент. Такие программы предоставляют ученику возможность наблюдать на экране дисплея некоторый процесс, влияя на его ход подачей команды с клавиатуры, меняющей значения параметров.

Программы 4-го типа (игры) предоставляют в распоряжение ученика некоторую воображаемую среду, существующий только в компьютере мир, набор каких-то возможностей и средств их реализации. Использование предоставляемых программой средств для реализации возможностей, связанных с изучением мира игры и деятельностью в этом мире. приводит к развитию обучаемого, формированию у него познавательных навыков, самостоятельному открытию им закономерностей, отношений объектов действительности, имеющих всеобщее значение.

Наибольшее распространение получили обучающие программы первых двух типов в связи с их относительно невысокой сложностью, возможностью унификации при разработке многих блоков программ. Если программы 3-го и 4-го типов требуют большой работы программистов, психологов, специалистов в области изучаемого предмета, педагогов-методистов, то технология создания программ 1-го и 2-го типов ныне сильно упростилась с появлением инструментальных средств или наполняемых автоматизированных обучающих систем (АОС).

Основные действия, выполняемые программами первых двух типов:

* предъявление кадра с текстом и графическим изображением;
* предъявление вопроса и меню вариантов ответа (или ожидание ввода открытого ответа);
* анализ и оценка ответа;
* предоставление кадра помощи при нажатии специальной клавиши.

Они могут быть легко и унифицировано запрограммированы, так что разработчику обучающей программы остается ввести в компьютер только соответствующий текст, варианты ответов, нарисовать на экране с помощью манипулятора “мышь” картинки. Создание обучающей программы в этом случае выполняется совершенно без программирования, не требует серьезных компьютерных познаний и по силам любому педагогу-предметнику средней школы.

## **1.3. Элементы управления в сценариях обучающих программ**

В соответствии с постулатами общей теории управления в любых циклических замкнутых системах управления, в том числе и в педагогических, должны быть реализованы следующие функции:

* формирование целей управления;
* установление исходного состояния объекта управления;
* определение программы воздействий, предусматривающей основные переходные состояния объекта управления;
* систематический сбор информации обратной связи;
* переработка информации обратной связи с целью выработки и реализации корректирующих воздействий.

Остановимся более подробно на особенностях понятия обратной связи, присущих педагогическим системам. Обратную связь (ОС) в триаде "Педагог - Обучающая программа - Обучаемый" можно разделить на два вида: внешняя и внутренняя ОС.

Внутренняя ОС - это информация, которая поступает от обучающей программы к ученику в ответ на его действия при выполнении упражнений. Она предназначена для самокоррекции учеником своей учебной деятельности. Понятие внутренней ОС имеет исключительно важное значение для автоматизации процесса обучения. Внутренняя ОС дает возможность ученику сделать осознанный вывод об успешности или ошибочности учебной деятельности. Она побуждает ученика к рефлексии, является стимулом к дальнейшим действиям, помогает оценить и скорректировать результаты учебной деятельности. Различают консультирующую и результативную внутреннюю ОС. Консультация может быть разной: помощь, разъяснение, подсказка, наталкивание и т.п. Результативная ОС также может быть различной: от "верно - неверно" до демонстрации правильного результата или способа действия.

Информация внешней ОС в рассматриваемой триаде поступает к педагогу и используется им для коррекции деятельности ученика и обучающей программы.

# **ГЛАВА 2. СРЕДСТВА, ИСПОЛЬЗУЕМЫЕ ПРИ СОЗДАНИЕ ОБУЧАЮЩЕЙ ПРОГРАММЫ**

## **2.1. Язык разметки гипертекста HTML и каскадные таблицы стилей CSS**

### **2.1.1. Общие сведения о HTML**

Hypertext Markup Language (HTML) это простая система для создания гипертекстовых документов, которые переносимы с одной платформы на другую. В сущности, HTML документы это SGML документы с общей семантикой, которая подходит для представления информации в большом количестве приложений. HTML может использоваться для представления:

* Гипертекстовых новостей, почты, документации и гипермедиа
* Меню опций
* Результатов запросов к базам данных
* Простых документов с встроенными графическими образами
* А также, для гипертекстового просмотра существующих массивов информации

Проект [World Wide Web](http://www.citycat.ru/doc/HTML/short/whatis3w.html) (WWW) объединяет разбросанную по миру информацию. Для этого WWW использует Internet [HyperText Transfer Protocol](http://www.citycat.ru/doc/HTML/short/http.html) (HTTP) — Протокол Передачи Гипертекста, который позволяет ввести соглашения на способы представления данных при взаимодействии клиента и сервера.

Данные передаются в теле сообщения в соответствии с MIME стандартом. HTML — это одно из представлений информации в WWW. Предполагается, что HTML соответствует одному из типов MIME, а именно text/html. Дополнительно, HTML это разработка, соответствующая Международному Стандарту ISO 8879 - Standard Generalized Markup Language (SGML), который является системой для определения структурированных типов документов.

Язык **HTML** - это основа web сайтов, с его помощью создается каркас страницы, которую видно в браузере.

Если сравнивать страницу сайта и обычную бумажную книгу, то на сайте, как и в книге, есть абзацы и заголовки. В книге есть название всей книги (по сути самый главный заголовок), есть названия глав, параграфов в этих главах и так далее.

Заголовки, абзацы и другие блоки можно выделить и на странице сайта. Это делается с помощью *HTML тегов*.

**HTML теги** - это специальные команды для браузера. Они говорят ему, что, к примеру, следует считать заголовком страницы, а что абзацем.

Теги строятся по такому принципу: уголок **<**, потом имя тега, а потом уголок **>**, вот так: **<имя тега>**. Имя тега может состоять из английских букв и цифр. Примеры тегов: **<h1>**, **<p>**, **<b>**. Вот ключевые:

* Основные: html, head, title, body.
* Структурные: div, span.
* Текстовые: p, ul, ol, li, h1-h6, br, em, strong, b, i.
* Таблицы: table, tr, td, th.
* Ссылки: a.
* Мультимедиа: img, object.
* Фреймы: frameset, frame, iframe.
* Формы: form, input, textarea, label, select, option.
* Факультативные: hr.
* Специальные: script, link, meta.

Теги обычно пишутся парами - открывающий тег и соответствующий ему закрывающий. Разница между открывающим и закрывающим тегами в том, что в закрывающем теге после уголка **<** стоит слеш **/**.

К примеру, **<p>** - так я открыл тег p, а так - **</p>** - я его закрыл. Все, что попадает между открывающим и закрывающим тегами, подпадает под воздействие нашего тега.

Бывают теги, которые не нужно закрывать, например, **<br>** или **<img>**.

В тегах также могут размещаться **атрибуты** - специальные команды, которые расширяют действие тега.

Атрибуты размещаются внутри открывающего тега в таком формате: **<тег атрибут1="значение" атрибут2="значение">**.

Кавычки могут быть любыми - одинарными или двойными, допустимо их вообще их не ставить, если значение атрибута состоит из одного слова (но это не желательно).

Cтраница сайта - это обычный текстовый файл с расширением **.html**. Внутри этого файла и хранится текст HTML страницы вместе с тегами. Этот файл обязательно должен иметь следующие теги: тег **<html>**, который должен содержать в себе текст всего сайта (все, что написано вне этого тега, браузером будет проигнорировано), а внутри него должны быть еще два тега: тег **<head>** для служебного содержимого страницы и тег **<body>** - для основного текста, который и виден на экране браузера.

В служебное содержимое, которое располагается внутри тега **<head>**, входит много различных вещей, но пока нам нужны только две из них. Это тег **<title>**, задающий название страницы, которое будет видно во вкладке браузера, и тег **<meta>**, который задает кодировку страницы (она ставится в атрибуте **charset** и обычно имеет значение **utf-8**).

Кроме того, перед тегом **<html>** обычно пишется конструкция **doctype**, которая указывает версию языка HTML, на которой сделан сайт. Актуальная сейчас версия языка имеет номер пять и доктайп для нее должен выглядеть так - **<!DOCTYPE html>**.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>***Это заголовок тайтл***</title>

</head>

<body>

***Это основное содержимое страницы.***

</body>

</html>

Одним из основных элементов страницы являются **абзацы**. Их можно сравнить с абзацами в книге - каждый абзац начинается с новой строки.

Абзац создается с помощью тега **<p>** таким образом:

<!DOCTYPE html>

<html>

<head>

<title>Это заголовок тайтл</title>

</head>

<body>

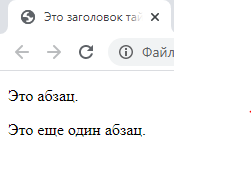
<p> Это абзац. </p>

<p> Это еще один абзац. </p>

</body>

</html>

Так код будет выглядеть в браузере (рис 2.1.):



***Рис 2.1. Пример в браузере***

Кроме абзацев важное значение на странице имеют **заголовки**. Их также можно сравнить с заголовками из книги - каждая глава имеет свой заголовок (название этой главы) и разбита на параграфы, которые тоже имеют свои заголовки. **Заголовки создаются** с помощью тегов **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>**, **<h6>**. Они имеют разную степень важности. В заголовке **h1** следует располагать **название всей HTML страницы**, в **h2** - название **блоков** страницы, в **h3** - название подблоков и так далее.

<!DOCTYPE html>

<html>

<head>

<title>Это заголовок тайтл</title>

</head>

<body>

<h1>Заголовок h1</h1>

<h2>Заголовок h2</h2>

<h3>Заголовок h3</h3>

<h4>Заголовок h4</h4>

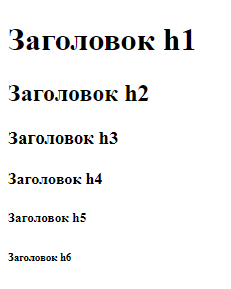
<h5>Заголовок h5</h5>

<h6>Заголовок h6</h6>

</body>

</html>

Так код будет выглядеть в браузере (рис 2.2.):



***Рис 2.2. Пример в браузере***

Наряду с абзацами и заголовками существует еще один важный элемент страницы - это **списки**. Такие элементы наверняка знакомы всем пользователям интернета. Они представляют собой перечисление чего-либо (список) по пунктам. Возле каждого пункта списка обычно стоит закрашенный кружок (он называется **маркер** списка).

Списки создаются с помощью тега **<ul>**, внутри которого обязательно должны идти теги **<li>**. Тег **ul** задает сам список, а в теги **li** следует ложить пункты списка (то есть одному li соответствует один маркер списка). Надо иметь ввиду, что отдельно теги li использовать нельзя.

<!DOCTYPE html>

<html>

<head>

<title>Это заголовок тайтл</title>

</head>

<body>

<ul>

<li>Первый пункт списка.</li>

<li>Второй пункт списка.</li>

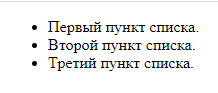
<li>Третий пункт списка.</li>

</ul>

</body>

</html>

Так код будет выглядеть в браузере (рис 2.3.):



***Рис 2.3. Пример в браузере***

Списки, созданные через тег **ul**, называются **неупорядоченными** списками. Такое название им дано потому, что существуют еще и **упорядоченные** списки, у которых вместо маркеров-точек будут маркеры-цифры. Такие списки вместо тега ul имеют тег **<ol>**, а пункты таких списков также создаются через теги **li**.

<!DOCTYPE html>

<html>

<head>

<title>Это заголовок тайтл</title>

</head>

<body>

<ol>

<li>Первый пункт списка.</li>

<li>Второй пункт списка.</li>

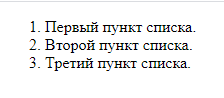
<li>Третий пункт списка.</li>

</ol>

</body>

</html>

Так код будет выглядеть в браузере (рис 2.4.):



***Рис 2.4. Пример в браузере***

Для размещения изображения предназначен тег **<img>**, имеющий обязательный атрибут **src**, в котором хранится путь к файлу картинки.

<!DOCTYPE html>

<html>

<head>

<title>Это заголовок тайтл</title>

</head>

<body>

<img src="mi171.jpg">

</body>

</html>

Так код будет выглядеть в браузере (рис 2.5.):



***Рис 2.5. Пример в браузере***

Практически во всех языках программирования есть такое понятие как "комментарии". **Комментарии** - это такой текст, который будет проигнорирован браузером - на экране он будет не виден, но останется в исходном коде страницы. Комментарии в HTML оформляются следующим образом: сначала уголок, знак ! и два дефиса - **<!--**, потом текст комментария, а потом два дефиса и уголок - **-->**.

<!DOCTYPE html>

<html>

<head>

<title>Это заголовок тайтл</title>

</head>

<body>

Содержимое страницы.

<!--

Это комментарий HTML -

на странице сайта он не отображается.

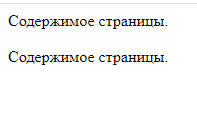
-->

Содержимое страницы.

</body>

</html>

Так код будет выглядеть в браузере (рис 2.6.):



***Рис 2.6. Пример в браузере***

### **2.1.2. Что из себя представляет CSS**

**CSS** — это язык стилей, определяющий внешний вид HTML-документов. Говоря о CSS подразумевается работа со шрифтами, полями, высотой, шириной и т.д. Проще говоря, CSS значительно расширяет возможности оформления сайтов.

Как сделать так, чтобы написанный CSS применился к **HTML коду**? Это можно сделать тремя способами.

**Первый** способ заключается в том, что **CSS** будет написан в отдельном файле и специальным образом подключен к нашей HTML странице.

**Еще два способа** заключаются в том, что **CSS** код будет написан прямо на HTML странице.

**Первый способ** используется **гораздо чаще**. Одновременно можно применять все три способа.

Самый распространенный способ подключения CSS - это когда он хранится в отдельном файле и этот файл подключается ко всем страницам нашего сайта.

Преимущество такого подхода в том, что CSS файл один, а HTML файлов может быть любое количество, хоть тысяча. Мы сделаем изменение в одном месте CSS файла, например, покрасим все абзацы в красный цвет, и эти изменения применятся на всей 1000 HTML страниц, к которым подключен наш CSS файл. Очень удобно и быстро.

Чтобы подключить CSS файл к HTML странице, в теге **head** следует написать такую конструкцию: *<link rel="stylesheet" type="text/css" href="style.css">*.

Сам CSS файл должен быть с расширением **.css**. Обычно его называют *styles.css* или *style.css*.

<!DOCTYPE html>

<html>

<head>

<title>Это заголовок тайтл</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

</body>

</html>

Каждому тегу в HTML соответствует так называемый **селектор** CSS. К примеру, тегу **<p>** соответствует CSS селектор **p**, с помощью которого мы можем обратиться ко всем абзацам HTML страницы и, например, покрасить их всех одновременно в красный цвет. После селектора следует ставить фигурные скобки **{}**, внутри которых следует писать **CSS свойства**.

**CSS свойства** и задают цвет, размер шрифта и другие интересные вещи. Их следует писать в таком формате: **имя** свойства, потом **двоеточие**, потом **значение** этого свойства (например, свойство - это цвет, а "синий" - это значение). Например:

Это HTML код:

<!DOCTYPE html>

<html>

<head>

<title>Это заголовок тайтл</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<p>

Абзац с текстом.

</p>

</body>

</html>

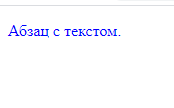
Это CSS код:

p {

color: blue;

}

Так код будет выглядеть в браузере (рис 2.7.):



***Рис 2.7. Пример в браузере***

Потом нужно поставить **точку с запятой** и можно писать следующее свойство.

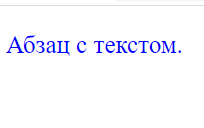
p {

color: blue;

font-size: 25px;

}

Так код будет выглядеть в браузере (рис 2.8.):



***Рис 2.8. Пример в браузере***

Если на какой-то странице сайта определены свои стили, которые перекрывают стили, описанные в .css файле, то на ней будут применены ее стили (т.е. определенные в заголовке этой страницы). В свою очередь, если на странице в заголовке определены какие-то стили, и существует стиль, указанный в тэге (например, <P STYLE= ".......">), то к данному тэгу будет применяться второй стиль.

## **2.2. Язык JavaScript**

Язык JavaScript предназначен для выполнения в браузере наряду с HTML и CSS. Но, если эти языки предназначены для верстки структуры сайта, то JavaScript позволяет 'оживлять' web-страницы - делать их реагирующими на действия пользователя или демонстрировать некоторую динамичность.

Написать и запустить JavaScript можно двумя способами: первый заключается в том, что код пишется прямо на HTML странице внутри тега <script>.

Тег <script> можно располагать в любом месте страницы - как в <head>, так и в <body>.

Второй вариант заключается в том, что JavaScript код хранится в отдельном файле (наподобие CSS) и подключается тоже с помощью тега <script> с атрибутом src, в котором указывается путь к файлу со скриптом.

Сегодня JavaScript может выполняться не только в браузере, но и на сервере или на любом другом устройстве, которое имеет специальную программу, называющуюся [«движком» JavaScript](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript).

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

Разные движки имеют разные «кодовые имена». Например:

* [V8](https://ru.wikipedia.org/wiki/V8_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript)) – в Chrome и Opera.
* [SpiderMonkey](https://ru.wikipedia.org/wiki/SpiderMonkey) – в Firefox.
* Ещё есть «Trident» и «Chakra» для разных версий IE, «ChakraCore» для Microsoft Edge, «Nitro» и «SquirrelFish» для Safari и т.д.

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания — что придаёт языку дополнительную гибкость.

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

* объекты с возможностью интроспекции;
* функции как объекты первого класса;
* автоматическое приведение типов;
* автоматическая сборка мусора;
* анонимные функции.

В языке отсутствуют такие полезные вещи, как:

* стандартная библиотека: в частности, отсутствует интерфейс программирования приложений по работе с файловой системой, управлению потоками ввода-вывода, базовых типов для бинарных данных;
* стандартные интерфейсы к веб-серверам и базам данных;
* система управления пакетами, которая бы отслеживала зависимости и автоматически устанавливала их.

В JavaScript:

* все идентификаторы регистрозависимы,
* в названиях переменных можно использовать буквы, подчёркивание, символ доллара, арабские цифры,
* названия переменных не могут начинаться с цифры,
* для оформления однострочных комментариев используются //, многострочные и внутристрочные комментарии начинаются с /\* и заканчиваются \*/.

На сегодняшний день поддержку JavaScript обеспечивают современные версии всех наиболее часто используемых браузеров. В Internet Explorer, Opera, Mozilla Firefox, Safari, Google Chrome, Яндекс Браузер имеется полная поддержка третьей редакции ECMA-262. При этом в Mozilla Firefox предпринята попытка осуществления поддержки четвёртой редакции спецификации, а первым браузером, в котором появилась неполная поддержка спецификации 3.1, явился Internet Explorer 8.

Допущенные разработчиками популярных браузеров ошибки в реализации спецификации, как правило, незначительны. По состоянию на ноябрь 2009 года объектная модель документа имеет более ограниченную поддержку.

По мнению создателя языка, поддержка в Internet Explorer компанией Microsoft одного из существующих и применяющихся в других браузерах быстрых движков JavaScript способно привести к появлению приложений, работающих с трёхмерной графикой, написанных на JavaScript 3D-игр, использованию JavaScript в задачах, в которых ранее применялась технология Adobe Flash.

Самыми простыми **типами данных** в JavaScript являются **строки** и **числа**.

**Числа** обозначают сами себя: 1, 12, 145, а вот **строки** требуется брать в кавычки (одинарные или двойные - без разницы).

**Имя переменной** должно состоять из английских букв: больших или маленьких, а также цифр и знака\_подчеркивания.

В JavaScript при **объявлении** переменной обязательно должно быть написано ключевое слово **var или let**:

***var*** student, student1, studentNumber, student\_number;

## **2.3. Bootstrap**

**Bootstrap — это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений.**

Основная область его применения – это разработка фронтенда сайтов и интерфейсов админок. Среди аналогичных систем (Foundation, UIkit, Semantic UI, InK и др.) фреймворк Bootstrap является самым популярным.

**Почему Bootstrap так популярен?** Это связано с тем, что он **позволяет верстать сайты в несколько раз быстрее**, чем это можно выполнить на «чистом» CSS и JavaScript. Сейчас это самый дорогой ресурс. Также его популярность связана с доступностью. Она заключается в том, что на нём даже начинающий разработчик может верстать достаточно качественные макеты, которые трудно было бы выполнить без глубоких знаний веб-технологий и достаточной практики.

Фреймворк Bootstrap представляет собой набор CSS и JavaScript файлов. Чтобы его использовать эти файлы необходимо просто подключить к странице. После подключения вам станут доступны инструменты данного фреймворка: колоночная система (сетка Bootstrap), классы и компоненты.

### **2.3.1. Из чего состоит Bootstrap**

Bootstrap состоит из:

* сетки;
* классов для стилизации текста, изображений, таблиц и другого контента;
* компонентов предназначенных для создания кнопок, различных форм на странице, горизонтальных и вертикальных навигационных меню, слайдеров, выпадающих списков, аккордеонов, модальных окон, всплывающих подсказок и других элементов интерфейса;
* классов для решения вспомогательных задач наиболее часто возникающими перед веб-разработчиками (выравнивание текста, скрытие или отображение элемента, задания цвета и фона элементу, задание margin и padding отступов, и др.).

Первое – это нужно научиться работать с сеткой. В Bootstrap она используется как для создания основного каркаса страницы, так и для разметки всего остального. Сетка в Bootstrap 3 построена на float, а Bootstrap 4 – на CSS Flexbox.

Следующее – это изучить компоненты и классы Bootstrap.

После изучение этих базовых вещей уже можно будет переходить к вёрстке сайта или веб-приложения.

### **2.3.2. Преимущества и недостатки Bootstrap**

Преимущества, которые даёт фреймворк Bootstrap при разработке на его основе frontend части сайтов и интерфейсов админок:

* высокая скорость создания качественной адаптивной вёрстки даже начинающими веб-разработчиками (достигается это благодаря использованию готовых компонентов, созданных профессионалами);
* кроссбраузерность и кроссплатформеннось (корректное отображение и работа сайта во всех поддерживаемых этим фреймворком браузерах и операционных системах);
* наличие большого количество готовых хорошо продуманных компонентов, протестированных огромным сообществом веб-разработчиков на различных устройствах;
* возможность настройки под свой проект, достигается это посредством изменения SCSS переменных и использования Bootstrap миксинов (можно изменить количество колонок, цвета, радиуса скруглений углов элементов, отступы между колонками и многое другое);
* низкий порог вхождения; для работы с фреймворком не обязательно иметь глубокие знания по HTML, CSS, JavaScript и jQuery (достаточно знать только основы вышеперечисленных технологий);
* наличие хорошо продуманного дизайна компонентов и согласованности (в Bootstrap все компоненты выполнены в едином стиле);
* наличие огромного сообщества, большого количества статей, рецептов и видеоматериалов; всё это при желании поможет не только хорошо разобраться в фреймворке, но и найти ответы практически на любые вопросы.

На Bootstrap, конечно, верстают далеко не всё. Его не используют, например, для:

* создания фронтенда проектов с уникальным дизайном;
* разработки проектов, в которых заказчик готов платить за проект на «чистом» CSS и JavaScript (в большинстве случаях такая разработка осуществляется в команде, в которой каждый её участник выполняет какой-то свой определенный набор функций);
* вёрстки личных проектов, если у вас есть достаточно количество времени и ваш уровень знания технологий HTML, CSS и JavaScript достаточный, чтобы это осуществить.

Bootstrap, так и большинство подобных фреймворков, имеет недостатки. Среди них:

* более большой размер конечных CSS и JavaScript файлов, чем если их написать конкретно под этот проект на чистом CSS и JavaScript (это связано с тем стили и код JavaScript содержат много всего и на большинстве сайтах нужна только та или иная часть из этого);
* использование Bootstrap для вёрстки сайтов, в которых потребуется значительное переписывание его CSS и JavaScript кода (простой настройкой Bootstrap переменных тут не обойтись).

Первый недостаток не такой существенный, тем более что его можно немного «сгладить» выполнив самостоятельную сборку проекта из исходных кодов и включить в неё только те компоненты, которые нужны. В этом случае код этого фреймворка будет значительно меньше.

### **2.3.3. Какую версию Bootstrap выбрать**

Для разработки большинства веб-проектов лучше использовать последнюю версию Bootstrap. На текущий момент - это 4.4.1.

Эта версия последняя и она основана, как было уже отмечено выше, на CSS Flexbox. Она поддерживает последние, стабильные версии всех основных браузеров и платформ. В операционной системе Windows эта версия поддерживает Internet Explorer 10-11 и Microsoft Edge.

Если же вам нужна поддержка более старых браузеров, например Internet Explorer 8 и 9, то в этом случае следует выбрать третью версию. Последняя актуальная версия из 3 ветки - это 3.4.1.

В старых версиях Internet Explorer (8 и ниже) некоторые компоненты Bootstrap 3 могут отображаться без градиентов, теней и закруглённых углов. Это связано с тем, что эти версии браузеров не поддерживают необходимые для этого CSS3 свойства.

## **2.4. React js**

React - это библиотека JavaScript, которая используется для создания пользовательского интерфейса. React был создан компанией Facebook, а первый релиз библиотеки увидел свет в марте 2013 года.

Первоначально React предназначался для веба, для создания веб-сайтов, однако позже появилась платформа React Native, которая уже предназначалась для мобильных устройств.

React представляется идеальный инструмент для создания масштабируемых веб-приложений (в данном случае речь идет о фронтенде), особенно в тех ситуациях, когда приложение представляет SPA (одностраничное приложение).

React относительно прост в освоении, имеет понятный и лаконичный синтаксис.

**2.4.1. Виртуальный DOM**

Вся структура веб-страницы может быть представлена с помощью DOM (Document Object Model)- организация элементов html, которыми мы можем манипулировать, изменять, удалять или добавлять новые. Для взаимодействия с DOM применяется язык JavaScript. Однако когда мы пытаемся манипулировать html-элементами с помощью JavaScript, то мы можем столкнуться со снижением производительности, особенно при изменении большого количества элементов. А операции над элементами могут занять некоторое время, что неизбежно скажется на пользовательском опыте. Однако если бы мы работали из кода js с объектами JavaScript, то операции производились бы быстрее.

Для решения проблемы производительности как раз и появилась концепция виртуального DOM.

Виртуальный DOM представляет легковесную копию обычного DOM. И отличительной особенностью React является то, что данная библиотека работает именно с виртуальным DOM, а не обычным.

Если приложению нужно узнать информацию о состоянии элементов, то происходит обращение к виртуальному DOM.

Если необходимо изменить элементы веб-страницы, то изменения вначале вносятся в виртуальный DOM. Потом новое состояние виртуального DOM сравнивается с текущим состоянием. И если эти состояния различаются, то React находит минимальное количество манипуляций, которые необходимы до обновления реального DOM до нового состояния и производит их.

В итоге такая схема взаимодействия с элементами веб-страницы работает гораздо быстрее и эффективнее, чем если бы мы работали из JavaScript с DOM напрямую.

## **2.4.2. Ключевые особенности  React: декларативность, универсальность, компонентный подход, виртуальный DOM, JSX**

Одна из ключевых особенностей React — универсальность. Эту библиотеку можно использовать на сервере и на мобильных платформах с помощью React Native. Это принцип Learn Once, Write Anywhere или «Научитесь один раз, пишите где угодно».

Ещё одна важная особенность библиотеки — декларативность. С помощью React разработчик описывает, как компоненты интерфейса выглядят в разных состояниях. Декларативный подход сокращает код и делает его понятным.

React основан на компонентах, это ещё одна ключевая особенность библиотеки. Каждый компонент возвращает часть пользовательского интерфейса со своим состоянием. Объединяя компоненты, программист создаёт завершённый интерфейс веб-приложения.

Важная особенность React — использование JSX. Это расширение синтаксиса JavaScript, которое удобно использовать для описания интерфейса. JSX похож на HTML, тем не менее это всё-таки JavaScript. Пример JSX можно увидеть ниже:

const header = text ? <h1>{text}</h1> : null;

const vdom = (

<div>

{header}

<Hello />

</div>

);

JSX позволяет писать JavaScript-код с помощью готовых компонентов, которые практически полностью повторяют HTML. Это упрощает разработку.

К важным особенностям React относится использование виртуального DOM (Virtual DOM). Виртуальный DOM — объект, в котором хранится информация о состоянии интерфейса. При изменении состояния, например, после отправки формы или нажатия кнопки, React рассчитывает разницу между старым и новым состоянием. После этого библиотека отрисовывает новое состояние. Использование виртуального DOM позволяет библиотеке эффективно обновлять реальный DOM.

Промежуточный итог: React — популярная библиотека для работы с UI. К её основным особенностям относится декларативность, компонентный подход, универсальность, использование виртуального DOM и JSX.

## **2.4.3. Рендеринг элемента в DOM**

Допустим, в вашем HTML-файле есть <div>:

<div id="root"></div>

Мы назовём его «корневым» узлом DOM, так как React DOM будет управлять его содержимым.

Обычно в приложениях, написанных полностью на React, есть только один корневой элемент. При встраивании React в существующее приложение вы можете рендерить во столько независимых корневых элементов, во сколько посчитаете нужным.

Для рендеринга React-элемента в корневой узел DOM вызовите [ReactDOM.render()](https://ru.reactjs.org/docs/react-dom.html" \l "render) с React-элементом и корневым DOM узлом в качестве аргументов:

const element = <h1>Hello, world</h1>;

ReactDOM.render(element, document.getElementById('root'));

На странице будет написано «Hello, world».

# **ГЛАВА 3. Разработка обучающей программы силовой установки вертолета (двигатель ТВ3-117ВМ)**

## **3.1. Постановка задачи**

Требуется разработать обучающую программу. Система должна содержать список категорий и под категорий, возможность просматривать материалы по дисциплине. Должна быть возможность смены языка в программе.

При разработке будут использованы следующие средства :

HTML(XML) - для разметки интерфейса ;

JavaScript – для функционала страниц;

CSS - для задания привлекательных стилей;

React js -для создания обучающей программы

## **3.2. Структура главной страницы**

Структура сайтов данного типа должна отличаться своей легкостью пользования и простотой. Структура данной работы (рис 3.1.) состоит из четырёх базовых составляющих :

Logo - логотип

Menu Bar – лента меню

Content – содержимое сайта

***Рис 3.1. Структура пользовательского интерфейса***

## **3.3. Создание главной страницы сайта**

Для создания обучающей программы были использованы средства Web-программирования такие как HTML, CSS, Bootstrap, JavaScript и React js.

С помощью React js создаются Single Page Application (SPA).

Главное отличие одностраничных приложений от традиционных многостраничных заключается в том, что навигация по одностраничному приложению не предполагает перехода на совершенно новую страницу. Вместо этого все страницы (называемые представлениями) обычно загружаются внутри одной и той же начальной страницы. Главная сложность в разработке SPA — сделать так, чтобы одностраничное приложение вело себя таким образом, как привыкли пользователи:

1. URL-адрес, отображаемый в адресной строке, должен всегда отображать то, что пользователь просматривает.
2. Пользователь ожидает, что сможет использовать кнопки истории браузера «Назад» и «Вперед».
3. Пользователь должен иметь возможность перейти к определенному представлению напрямую, по URL-адресу.

Чтобы решить все эти проблемы, существует множество библиотек JavaScript, самая главная из которых — React Router.

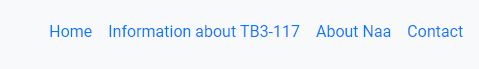
Главная страница состоит из Header, Content place, Footer.

В **Header** находятся:

Лого



Навигация по приложению



Элемент для смены языка в приложении.



**Коды**:

import React, { Component, useState } from 'react'

import posed from 'react-pose';

import "./Navbar.css"

import HamburgerMenu from 'react-hamburger-menu'

import {Link} from 'react-router-dom';

import Select from 'react-select';

import { useTranslation } from 'react-i18next';

import headerLinks from "../../data/header.json";

const optionsLang = [

    { value: 'en', label: 'EN' },

    { value: 'ru', label: 'RU' },

    { value: 'az', label: 'AZ' }

  ]

const Animation = posed.div({

    closed: { height: 0 },

    open: { height: 'auto' }

})

const Navbar=(props)=>{

    const { history } = props

    const [open, setOpen] = useState(false);

    const { t, i18n } = useTranslation(['']);

    const lang=i18n.language;

   const handleClick=()=> {

    setOpen(!open);

    }

   const  NavClick=()=>{

        setOpen( false);

        window.scrollTo(0, 0);

    }

        return (

            <nav className='navbar-nav navbar navbar-expand-lg navbar-darg bg-light mb-3 p-3 headMain'>

            <Link to='/' className='navbar-brand'>

                <img className="img" src="https://naa.az/wp-content/uploads/26637838\_1503945446356288\_1034795990\_n.png" width="200px" alt="Logo"/>

            </Link>

            <Animation className="animation ml-auto" pose={open? 'open' : 'closed'}>

            <ul className='navbar-nav ml-auto ul'>

                {headerLinks[lang].map((item,index)=>(

                    <li className='nav-item active' onClick={NavClick} key={index}>

                        <Link to={item.to} className='nav-link'>{item.title}</Link>

                    </li>

                ))}

                <li className="nav-item active lang">

                </li>

            </ul>

            </Animation>

            <Select

          placeholder={lang.toUpperCase()}

          isSearchable ={false}

          defaultValue ={lang.toUpperCase()}

          onChange={(item)=>{

          let val=item.value;

        //   i18n.changeLanguage(val);

        // console.log(window.location.pathname.substr(0));

          let number = 0;

                headerLinks[val].map((item, index) => {

                  if (window.location.pathname.substr(0) !== '/' && item.to ==

                    window.location.pathname.substr(0) || window.location.pathname.substr(3) !== '/' && item.to ==

                    window.location.pathname.substr(3)) {

                    number = index;

                  }

                })

                window.location.pathname = val + headerLinks[val][number].to;

          // window.location.pathname = i18n.language;

        }} options={optionsLang} />

        <span  className='span-ham'>

            <HamburgerMenu

            className='ham'

            isOpen={open}

            menuClicked={handleClick}

            width={35}

            height={25}

            strokeWidth={3}

            rotate={0}

            color='black'

            borderRadius={0}

            animationDuration={0.5}

            />

            </span>

        </nav>

        )

}

export default Navbar;

В **Content place** находятся слайдер с фотографиями двигателя и некоторая информация о двигателе.



**Коды** для Content на главной странице:

import React, { Component } from 'react'

import Slider2 from '../Slider/Slider2'

import "./welcome.css"

import { useTranslation } from 'react-i18next';

 const WelcomePage=()=>{

  const { t, i18n } = useTranslation(['home']);

    return (

        <div className="container" style={{marginBottom:100}}>

            <div className="row">

            <div className="col-12" style={{textAlign:"center",margin:"30px 0"}}>

              <h2>{t('title')} </h2>

            </div>

                <div className="col-12 d-flex justify-content-center" style={{textAlign:"center"}}>

                   <Slider2/>

                </div>

                <div className="col-12">

                <p>{t('content')}</p>

                <div className="col-12 d-flex justify-content-center">

                <img src="http://www.airwar.ru/image/i/engines/tv3-117bma.jpg" alt="photo-1" />

                </div>

                </div>

            </div>

        </div>

    )

}

export default WelcomePage;

В **Footer** находятся ссылки на Соц. Сети и сайт Национальной Академии Авиации и информация о студенте который разработал это приложение, то есть обо мне.



**Коды**:

import React, { Component } from 'react'

import "./footer.css";

import socData from "../../data/socialPage.json";

import { useTranslation } from 'react-i18next';

const Footer=()=> {

  const { t, i18n } = useTranslation(['footer']);

        return (

        <div className='navbar-nav navbar navbar-expand-lg navbar-darg bg-dark text-white mb-3 p-3 footMain'>

           <div className="col-md-4 col-12 soc">

           <a href={socData.facebook} target="\_blank" className="btn-face m-b-20">

              <i className="fab fa-facebook-f"/>

            </a>

            <a href={socData.globe} target="\_blank" className="btn-globe">

              <i className="fas fa-globe"/>

            </a>

            <a href={socData.instagram} target="\_blank" className="btn-instagram">

              <i className="fab fa-instagram"/>

            </a>

            </div>

           <div className="col-md-8 col-12 cred">

            {t('cr')} Salmanov Elvin 1457R

           </div>

        </div>

        )

}

export default Footer;

## **3.4.Каркас(Шаблон) для Страниц**

Header появляется на каждой странице, так же как и Footer. Чтобы на каждой странице не добавлять коды для Header и Footer, был создан Layout (каркас/шаблон), в котором находятся:

Header

…

Footer

Три точки означают, что на их месте будут коды, при использование Layout коды записаны, так чтобы компилятор принимал их как элемент который заменяет три точки.

Layout:

import React from "react";

import Header from "../Navbar/Navbar";

import Footer from "../footer/footer";

export default function LayoutOne(props) {

  return (

    <>

      <Header />

        {props.children}

      <Footer />

    </>

  );

}

## **3.5. Просмотр информации о двигателе**

Для того чтобы просмотреть информацию о двигателе, нужно парейти по ссылку “[Information about ТВ3-117](https://tb3maa.vercel.app/en/info)” , если приложение на англ. языке, если же на других языках то соответствующий текст на выбранном языке. Откроется страница на которой размешены Header,Breadcrumbs, Information about TB3-117 и Footer.

**Breadcrumbs** - это элемент навигации по веб-сайту, представляющий собой путь по сайту от его "корня" до текущей страницы, на которой находится пользователь.



**Коды:**

import React from "react";

import { Link } from 'react-router-dom';

import "./Breadcrumbs.css";

export const BreadcrumbItem = ({ name, current, href }) => {

  return (

    <li className="li">

      {href ? <Link className="breadCr" to={href}>{name}</Link>

        : <a className="breadCr">{name}</a>

      }

    </li>

  )

};

export const Breadcrumb = ({ title, children }) => {

  return (

    <div className="breadcrumb">

      <div className="container">

        <h2> {title}</h2>

        <ul>{children}</ul>

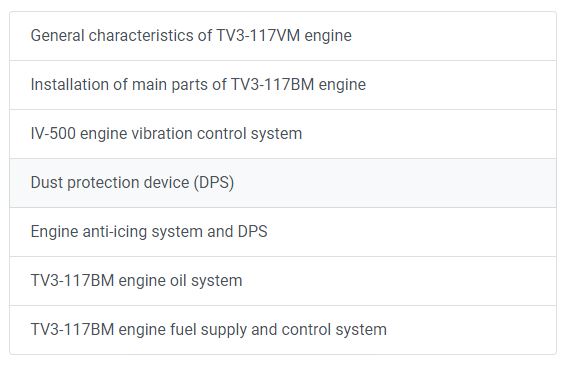
      </div>

    </div>

  );

};

Разделы Information about TB3-117:



Названия разделов берутся из json файла, как и другие некоторые данные в приложении.

{"az":[

    { "title": "ТВ3-117ВМ müherrikin ümumi xarakteristikası", "to": "/info/xarakteristika"},

    { "title": "ТВ3-117ВМ müherrikin əsas hissələrin qurğusu", "to": "/info/esashisselerinqurgusu"},

    { "title": "ИВ-500 mühərrikin titrəmə nəzarəti sistemi", "to": "/info/vibrasiya"},

    { "title": "Tozqoruyucu qurğu(TQQ)", "to": "/info/pzu"},

    { "title": "Mühərrikin buzlanma ələyhinə sistemi və TQQ", "to": "/info/buzlamaeleyhi"},

    { "title": "ТВ3-117ВМ müherrikin yağ sistemi", "to": "/info/yaglama"},

    { "title": "ТВ3-117ВМ müherrikin yanacağla qidalanma və tənzimlənmə sistemi", "to": "/info/yanacaglaqidalanmavetenzimlenme"}

  ],

  "en":[

    { "title": "General characteristics of TV3-117VM engine", "to": "/info/characteristic"},

    { "title": "Installation of main parts of TV3-117ВМ engine", "to": "/info/assembly"},

    { "title": "IV-500 engine vibration control system", "to": "/info/vibration"},

    { "title": "Dust protection device (DPS)", "to": "/info/pzu"},

    { "title": "Engine anti-icing system and DPS", "to": "/info/antiIcing" },

    { "title": "TV3-117ВМ engine oil system", "to": "/info/oil"},

    { "title": "TV3-117ВМ engine fuel supply and control system", "to": "/info/fuelfeedingandadjustment"}

  ],

  "ru":[

    { "title": "Общие характеристики двигателя ТВ3-117ВМ", "to": "/info/xarateristiki"},

    { "title": "Установка основных частей двигателя ТВ3-117ВМ", "to": "/info/ustanovkaosnovnixcastey"},

    { "title": "Система контроля вибрации двигателя ИВ-500", "to": "/info/vibraciya"},

    { "title": "Устройство защиты от пыли (ПЗУ)", "to": "/info/pzu" },

    { "title": "Система антиобледенения двигателя и ПЗУ", "to": "/info/antiobledeneniye"},

    { "title": "Масляная система двигателя ТВ3-117ВМ", "to": "/info/maslo"},

    { "title": "Система подачи топлива и управления двигателем ТВ3-117ВМ", "to": "/info/podacatoplivaiupravleniye"}

  ]

}

В зависимости от выбранного языка в приложении, отображаются названия разделов на выбранном языке.

Если перейти по какой то из ссылок, то будет показываться информация про выбранную тему.

## **3.6. Страница с информацией о Национальной Академии Авиации**

Как и остальные страницы, эта страница состоит и Header, Breadcrumbs, Content и Footer.

Content состоит из информации о НАА, фотографии, и информации о руководстве и слайдера, состоящего из фотографий НАА.

import React, { Component } from 'react'

import Sli from "../Slider/Slider";

import { Breadcrumb, BreadcrumbItem } from "../Breadcrumbs/Breadcrumbs";

import "./About.css";

import { useTranslation } from 'react-i18next';

const About=()=> {

    const { t, i18n } = useTranslation(['about']);

        return (

<>

<Breadcrumb title={t('title')}>

        <BreadcrumbItem name={t('home')} href="/" />

        <BreadcrumbItem name={t('title')} current />

</Breadcrumb>

<div className="container">

<div className="row">

    <div className="col-12 col-md-6">

    <p>

    {t('content1')}

<br/> <br/>  {t('content2')}</p>

    </div>

    <div className="col-12 col-md-6">

    <img src="https://aviateka.su/wp-content/uploads/2018/03/naa-az-1024x709.jpg" alt="photo-1" width="100%" />

    </div>

</div>

<div className="row">

    <div className="col-12">

<p>

{t('content3')}

<br/>

{t('content4')}

<br/>

</p>

</div>

<div className="col-12 slider" >

    <Sli/>

</div>

<div className="col-12">

<p>

{t('content5')}

</p>

<br/>

<div className="col-12" style={{textAlign:"center",marginBottom:30}}>

<h2>

{t('boss')}

</h2>

</div>

<hr/>

<br/>

<div className="col-12" style={{marginBottom:30}}>

    <ul>

        <li>{t('boss1')}</li>

        <li>{t('boss2')}</li>

        <li>{t('boss3')}</li>

        <li>{t('boss4')}</li>

        <li>{t('boss5')}</li>

        <li>{t('boss6')}</li>

    </ul>

</div>

    </div>

</div>

</div>

</>

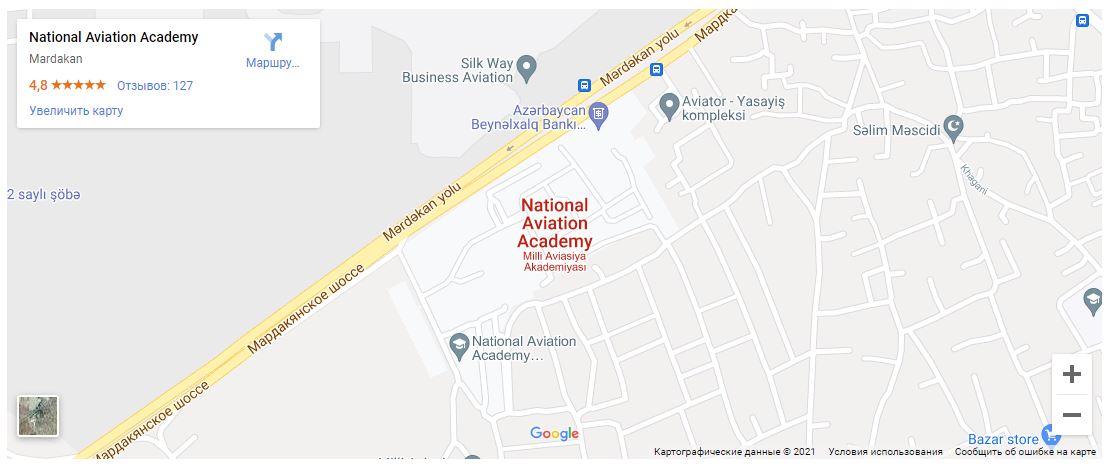
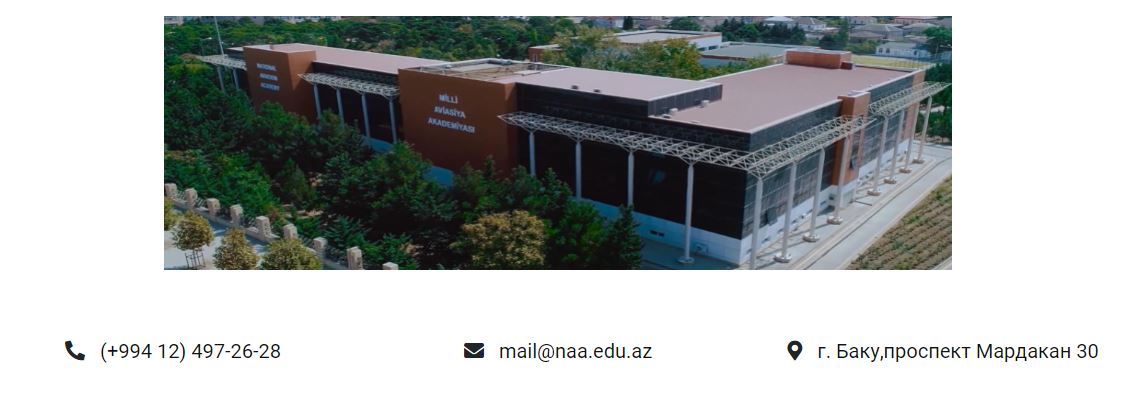
        )

}

export default About;

## **3.7. Страница с контактными данными НАА**

На этой странице содержатся контактные данные и местоположение НАА.



Перейти на сам сайт можно по ссылке:

<https://tb3maa.vercel.app/en/>

Исходный код доступен по ссылке:

<https://github.com/ElvinSalman/tb3naa>

# **ЗАКЛЮЧЕНИЕ**

При выполнении выпускной работы была поставлена задача: разработать обучающую программу по дисциплине **«Силовая установка вертолета (двигатель ТВ3-117ВМ)»**.

В ходе выполнения выпускной работы достигнуты следующие цели:

* Изучены некоторые технологии для разработки обучающей системы;
* Рассмотрена информация о двигателе TB3-117BM;
* Разработана обучающая программа;

В итоге подсистема обучения состоит из 32 категорий, которые содержат лекции по дисциплине.

# **ЛИТЕРАТУРА**

1. http://www.ito.su/2001/ito/II/II-4-7.html

Технология разработки и использования электронных учебников. Калинин Илья Александрович (МГПУ г.Москва)

1. http://www.btek.ru/dir/doc/nbp.ppt

Электронные учебные материалы. Нормативная база. Принципы и требования. Янголова Наталия Геннадьевна

1. Белоконь О.А., Белоконь Т.В. и др. Методические рекомендации по созданию учебно-методических комплексов для электронных обучающих систем
2. *Фримен Эрик, Фримен Элизабет.* Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. — П.: «Питер», 2010.
3. *Эд Титтел, Джефф Ноубл.* HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition. — М.: «Диалектика», 2011.
4. *Стивен Шафер.* HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2010.
5. *Jake Spurlock.* Bootstrap. Responsive Web-Development. — O'Reilly, 2013.
6. *David Cochran, Ian Whitley.* Bootstrap Site Blueprints. — Packt, 2014.
7. *Matt Lambert.* Bootstrap Site Blueprints. Volume II — Packt, 2016.
8. *Syed Fazle Rahman.* Jump Start Bootstrap. — SitePoint, 2014.
9. <https://www.w3schools.com/>
10. <https://ru.reactjs.org/docs/getting-started.html>