Министерство образования Нижегородской области

Государственное бюджетное профессиональное образовательное учреждение «Семеновский индустриально-художественный техникум»

Центр цифрового образования детей «IT-Куб»

СОГЛАСОВАНО:
Начальник ЦЦОД «IT-Куб»
_____ А.Н. Дубовицкий
«13» августа 2026 г.

УТВЕРЖДАЮ:
Директор ГБПОУ «СИХТ»
_____ И.Е. Напылов
Приказ № 274-од от «14» 08 2026 г.

ДОПОЛНИТЕЛЬНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ОБЩЕРАЗВИВАЮЩАЯ ПРОГРАММА

«WEB-РАЗРАБОТКА»

Направленность – техническая

Возраст обучающихся: 13-18 лет

Срок реализации: 1 год

Объем: 136 часов

Автор-составитель:

Красников Илья

г. Семенов, 2026 г.

Содержание

1. Комплекс основных характеристик программы 5

1.1. Пояснительная записка 5

1.2. Цели и задачи 11

1.3. Содержание общеразвивающей программы 13

1.3.1. Учебный план на 2026-2027 учебный год 13

1.3.2. Содержание учебного плана 19

1.4 Планируемые результаты 41

2.Комплекс организационно-педагогических условий реализации общеразвивающей программы 43

2.1. Календарный учебный график на 2025-2026 учебный год 43

2.2. Условия реализации программы 44

2.3. Формы аттестации и оценочные материалы 45

2.4. Методические материалы 50

Список литературы 51

1. Комплекс основных характеристик программы

1.1. Пояснительная записка

Программа «WEB-разработка» имеет техническую направленность. В ходе обучения обучающиеся знакомятся с современными инструментами фронтенд-разработки и приобщаются к инженерно-техническим знаниям в области информационных технологий. Освоение курса способствует формированию логического и технического мышления, развитию навыков проектной деятельности.

Программа объединяет достижения сразу нескольких направлений: в процессе обучения веб-разработке обучающиеся получают дополнительные знания в области информатики и алгоритмизации, основ дизайна и пользовательского интерфейса, а также расширяют словарный запас в области технического английского языка.

Программа «WEB-разработка» позволяет получить обучающимся необходимый объём знаний и навыков в зависимости от уровня подготовки и индивидуальных интересов. Она рассчитана на обучающихся 13–18 лет и предусматривает постепенное усложнение материала — от основ HTML и CSS до создания собственного веб-проекта с использованием React и защиты его на итоговом занятии.

Программирование занимает одну из значительных ниш в современном мире. Нынешнее программирование многогранно и используется в таких важных сферах как строительство, бизнес и экономика, медицина, биология и физика. Большой процент физического труда в промышленности заменен на машинный и роботизированный труд, который управляется посредством программного обеспечения, что обеспечивает существенный прирост скорости, точности операций и эффективности производства. Такое богатство разнообразия применений обеспечивается солидным выбором языков программирования, у каждого из которых есть свои плюсы и минусы.

Дополнительная общеобразовательная программа «WEB-разработка» имеет техническую направленность и делает акцент на **фронтенд-**

разработке – создании интерфейсов веб-сайтов. В отличие от традиционных курсов программирования, где упор делается на алгоритмы и консольные приложения, данный курс ориентирован на визуальный результат – обучающиеся сразу видят, как их код формирует веб-страницы в браузере. Новизна программы заключается во включении современных технологий (React, SPA) уже на начальном уровне обучения. Обучающиеся не только осваивают классические основы верстки и скриптов на стороне клиента, но и знакомятся с концепцией одностраничных приложений (Single Page Applications), состоянием приложения и работой с внешними данными через API. Таким образом, программа сочетает фундаментальные знания с современными подходами веб-разработки, что повышает мотивацию обучающихся.

Основанием для проектирования и реализации данной общеразвивающей программы служит перечень следующих нормативных правовых актов:

- ✓ Конституция Российской Федерации (принята всенародным голосова-ни-ем 12.12.1993 с изменениями, одобренными в ходе общероссийского голосования 01.07.2020);
- ✓ Указ Президента Российской Федерации от 09.11.2022 №809 «Об утверждении Основ государственной политики по сохранению и укреп-лению традиционных российских духовно-нравственных ценностей»;
- ✓ Указ Президента Российской Федерации от 07.05.2024 №309 «О национальных целях развития Российской Федерации на период до 2030 года и на перспективу до 2036 года»;
- ✓ Федеральный закон от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации»;
- ✓ Федеральный закон от 24.07.1998 № 124-ФЗ «Об основных гарантиях прав ребенка в РФ»;
- ✓ Постановление Правительства РФ от 26 декабря 2017 г. N 1642 "Об утверждении государственной программы Российской Федерации "Развитие образования"

- ✓ Концепция развития дополнительного образования до 2030 года (утверждена распоряжением Правительства РФ от 31.03.2022 № 678-р); -
- ✓ Распоряжение Правительства Нижегородской области от 12.09.2022 № 1057-р "О мерах по реализации Концепции развития дополнительного образования детей до 2030 года в Нижегородской области"
- ✓ Приказ Министерства просвещения РФ от 27 июля 2022 г. N 629 "Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам"
- ✓ Постановление Правительства РФ от 11.10.2023 N 1678 «Об утверждении правил применения организациями, осуществления образовательную деятельность, электронного обучения, дистанционных образовательных технологий при реализации образовательных программ»;
- ✓ Профессиональный стандарт «Педагог дополнительного образования детей и взрослых» (приказ Министерства труда и социальной защиты РФ от 22 сентября 2021 г. № 652н «Об утверждении профессионально-го стандарта «Педагог дополнительного образования детей и взрос-лых»);
- ✓ Методические рекомендации по созданию и функционированию центров цифрового образования «IT-куб» (утверждены письмом Министерства просвещения Российской Федерации от 30 ноября 2023 г. N АЗ-1750/04);
- ✓ Письмо Минпросвещения России от 29.09.2023 N АБ-3935/06 "О методических рекомендациях" (вместе с "Методическими рекомендациями по формированию механизмов обновления содержания, методов и технологий обучения в системе дополнительного образования детей...";
- ✓ Письмо Министерства образования и науки РФ от 18.11.2015г. № 093242. «О направлении Методических рекомендаций по проектированию дополнительных общеразвивающих программ (включая разноуровневые)»;

- ✓ Санитарные правила СП 2.4.2.4283-26 "Санитарно-эпидемиологические требования к организациям воспитания и обучения, отдыха и оздоровления детей и молодежи" .
- ✓ локальные акты образовательной организации.

Направленность заключается в предоставлении участникам современных знаний и навыков в области создания и дизайна веб-сайтов и веб-приложений. Основная цель программы — сформировать у обучающихся способность создавать качественные и функционально насыщенные веб-проекты, развивать творческую инициативу и готовность к решению профессиональных задач в сфере ИТ.

Ключевые компоненты направленности программы:

Практико-ориентированное обучение: Акцент делается на приобретении прикладных навыков, начиная с изучения HTML, CSS и JavaScript, заканчивая созданием реальных веб-проектов.

Развитие творческих способностей: Участники осваивают принципы современного дизайна, развивая эстетическое чувство и внимание к деталям, необходимым для качественной веб-разработки.

Алгоритмизация и логика: Через практику создания веб-приложений развивается умение мыслить структурированно, планировать ход действий и решать возникающие задачи последовательно.

Информационная культура: Формируются компетенции по грамотному поиску необходимой информации, выбору надежных источников и правильному применению найденных сведений.

Коммуникационные навыки: Участникам предоставляется возможность обмениваться опытом, сотрудничать в команде и презентовать свои работы, формируя важные социальные и профессиональные компетенции.

Данная программа рассчитана на широкий круг лиц, интересующихся современной цифровой средой и желающих приобрести востребованные профессии будущего.

Актуальность программы обусловлена тем, что современную эпоху цифровых технологий навыки веб-разработки являются востребованными и полезными для школьников. Умение создавать адаптивные веб-сайты, работать с интерактивными элементами и подключаться к внешним данным через API способствует формированию у подростков цифровой грамотности и расширяет их возможности для дальнейшего обучения и карьеры в сфере ИТ. Программа отвечает запросам времени, поскольку интернет-технологии проникли во все сферы жизни, а базовые знания HTML, CSS, JavaScript и современных фреймворков (таких как React) становятся частью общего технического кругозора молодёжи.

Веб-разработка — это направление создания сайтов и веб-приложений, основанное на открытых стандартах и широко доступных инструментах. Она объединяет разметку **HTML**, стили **CSS** и программирование на **JavaScript**, а также современные библиотеки (например, **React**) для построения удобных и быстрых интерфейсов. Веб-разработку используют сотни тысяч специалистов по всему миру для создания одностраничных приложений (SPA), подключения к внешним данным через **API**, адаптивной верстки под разные устройства и обеспечения доступности для всех пользователей. Освоение этого направления стимулирует интерес и любознательность, развивает способность решать проблемные ситуации, исследовать задачу, анализировать ресурсы, выдвигать и проверять идеи, планировать решения и воплощать их в работающем продукте. Кроме того, учебные проекты в вебе естественно развивают коммуникативные навыки обучающихся благодаря активному взаимодействию в командах: распределению ролей, совместной разработке, код-ревью и публичной защите результата.

В то же время объективные процессы информатизации российского общества формируют социальный заказ в сфере образования в общем, и в сфере дополнительного образования, в частности, на увеличение внимания к информационной грамотности обучающихся. Поэтому в структуру предлагаемой программы включены теоретический материал, практические задачи,

направленные на формирование начальной компьютерной грамотности, и информационной культуры, начальных навыков использования компьютерной техники, и современных информационных технологий для решения учебных и практических задач.

Прогностичность дополнительной общеобразовательной общеразвивающей программы «WEB-разработка» заключается в том, что она отражает требования и актуальные тенденции как сегодняшнего, так и завтрашнего дня цифровой экономики. Программа носит междисциплинарный характер: сочетает основы информатики, дизайна, математики, английского языка и коммуникации, что соответствует современным подходам к построению дополнительных образовательных программ и образования в целом.

Дополнительная общеобразовательная общеразвивающая программа «WEB-разработка» вовлекает обучающихся в осознанный процесс саморазвития и проектной деятельности. В ходе освоения курса школьники получают дополнительные знания в области информатики и алгоритмизации, основ графического и веб-дизайна, а также технического английского языка. Это обеспечивает комплексное развитие и формирует компетенции, востребованные в самых разных сферах — от образования и науки до бизнеса и творческих индустрий.

Отличительная особенность программы

Дополнительная общеобразовательная общеразвивающая программа «WEB-разработка» сочетает изучение теоретических основ построения веб-страниц и интерфейсов с получением практических навыков создания сайтов с использованием HTML, CSS, JavaScript и библиотеки React. Обучающиеся не только знакомятся с принципами семантической верстки и адаптивного дизайна, но и осваивают работу с динамическими данными и построение простых одностраничных приложений (SPA).

На обучение по данной программе принимаются дети в возрасте 13–18 лет. Такой возрастной диапазон выбран с учётом психолого-педагогических особенностей подростков, их готовности к освоению логики программирова-

ния и веб-технологий, а также исходя из уже имеющихся знаний и умений, полученных в общеобразовательных учреждениях по информатике и ИКТ.

Адресат общеразвивающей программы дополнительная общеобразовательная общеразвивающая программа «WEB-разработка» предназначена для обучающихся в возрасте 13-18 лет, не имеющих ограниченных возможностей здоровья, проявляющих интерес к программированию. Группа формируется по возрасту 13-18 лет.

Объем общеразвивающей программы 136 академических часов.

Формы обучения очная, с применением дистанционных образовательных технологий.

Срок освоения программы определяется содержанием программы и составляет 1 учебный год (136 академических часов).

Режим занятий длительность одного занятия составляет 2 академических часа (по 45 минут) с перерывом (переменой) в 10 минут для проветривания помещения и отдыха обучающихся. Периодичность занятий – 2 раза в неделю в соответствии с расписанием.

Педагогическая целесообразность программы «WEB-разработка» заключается в том, что она отвечает всем современным требованиям технологического образования, которое становится необходимым условием развития личности и подготовки к профессиональной деятельности. Современный этап характеризуется повсеместным внедрением цифровых технологий во все сферы жизни, и умение создавать веб-сайты и работать с интерфейсами становится базовым навыком. Поэтому раннее вовлечение подростков в техническое творчество через изучение веб-технологий является актуальным и соответствует интересам обучающихся этой возрастной группы, их способностям и возможностям.

Место проведения занятий: 606650, Нижегородская область, г. Семенов, ул. 1-е Мая, д. 27, каб № 304.

1.2 Цели и задачи курса

Целью программы является создание условий для развития интеллектуальных способностей и творческого потенциала учащихся, направленных на формирование навыков веб-разработки и умения применять их на практике при создании современных веб-сайтов. Курс нацелен на освоение учащимися основ технологий HTML, CSS, JavaScript и React, развитие алгоритмического мышления, а также на воспитание интереса к информационным технологиям и командной работе.

Задачи программы:

***Обучающие*:**

- обеспечить усвоение учащимися базовых теоретических знаний и практических умений в области веб-разработки;
- изучить основные технологии создания веб-страниц;
- научить принципам семантической верстки;
- освоить методы адаптивной верстки;
- дать базовые понятия о алгоритмизации и программировании на примере JavaScript;
- научить применять JavaScript для оживления веб-страниц;
- познакомить с работой с внешними данными через API;
- ввести учащихся в мир современных библиотек и фреймворков на примере React;
- научить выполнять сборку и запуск приложения;
- сформировать умения самостоятельно решать технические задачи;
- стимулировать мотивацию учащихся к получению знаний, помогать формировать творческую личность обучающегося.

***Развивающие*:**

- развитие у обучающихся интереса к программированию;
- развитие умения применять идеи программирования в решении конкретных задач;

- развитие коммуникативных компетенций: навыков сотрудничества в коллективе, малой группе (в паре), участия в беседе, обсуждении;
- формирование и развитие информационных компетенций: навыков работы с различными источниками информации, умения самостоятельно искать, извлекать и отбирать необходимую для решения учебных задач информацию.

Воспитательные:

- формирование самостоятельности и творческого подхода к решению задач с использованием средств вычислительной техники;
- воспитание упорства в достижении результата;
- способствование воспитанию личностных качеств: целеустремленности, настойчивости, самостоятельности, чувства коллективизма и взаимной поддержки, чувство такта;
- формирование коммуникативных качеств через коллективную творческую деятельность и реализацию коллективных проектов;
- формирование адекватной самооценки через проявление себя в выставках, конкурсах, мероприятиях, подведении итогов деятельности;
- формирование устойчивого интереса к выбранному виду деятельности.
- патриотическое воспитание через изучение истории страны и развития науки и техники во время занятий творчеством;
- формирование нравственных смыслов и духовных ориентиров;
- формирование у учащихся гражданской ответственности и правового самосознания, основанного на уважении к закону, знании прав человека и умении найти пути решения жизненных проблем;
- воспитание целеустремлённости, настойчивости и трудолюбия путём достижения учебных целей и подготовки к конкурсам и мероприятиям;
- воспитание доброго, бережного и качественного отношения к технике, взаимного уважения со своими товарищами, любви к трудовой деятельности.

1.3Содержание общеразвивающей программы

1.3.1 Учебный план

Таблица 1

№, п/п	Название раздела, темы	Описание тем	Кол-во часов		
			Всего	Теория	Практика
Модуль 1. Введение в веб-разработку			**4**	**1**	**3**
1-2	Знакомство, командообразование, инструктаж по ТБ и поведению в учебном кабинете. Старт курса и установка инструментов	Знакомство, командообразование, инструктаж по ТБ и поведению в учебном кабинете. Правила работы с компьютером. Входная диагностика в форме опроса. Цифровая гигиена. Знакомство с принципами работы веб-сайтов. Установка Node.js, Git и VS Code. Правила безопасной работы в сети.	2	1	1
3-4	Первый проект	Создаём репозиторий Git, оформляем README и пишем первый HTML-документ. Запускаем страницу в браузере.	2	-	2
Модуль 2. Основы разметки веб-страниц			**12**	**5**	**7**
5-6	Семантическая разметка HTML5: смысловые теги контента	Изучение заголовков h1–h6, абзацев, списков, ссылок и изображений. Разъяснение, почему семантика лучше, чем div. База доступности: alt и иерархия заголовков. Работа с заголовками и семантикой.	2	1	1
7-8	Семантическая структура страницы: области	Разметка header, nav, main, section, article, aside, footer. Одно main на страницу, правильная иерархия заголовков. Навигация по странице с помощью якорей. Написание семантической структуры.	2	1	1
9-10	Медиа-контент и подписи	Изучение добавления изображений и подписей через figure/figcaption. Разбор авторских прав и ссылок на источники.	2	1	1

11-12	Таблицы данных в HTML	Изучение работы с таблицами. Знакомство с тегами thead, tbody, tfoot и атрибутом scope у th. Создание таблиц на странице.	2	1	1
13-14	Формы: поля ввода и доступность	Изучение доступности для всех пользователей. Создание формы обратной связи: label, типы input, обязательные поля.	2	1	1
15-16	Практика по модулю 2	Выполнение практического задания по модулю 2.	2	-	2
Модуль 3. Оформление и макетирование страниц			**20**	**8**	**12**
17-18	Основы CSS: каскад и блочная модель	Разбор каскада, специфичности и блочной модели. Настройка отступов, рамок и проверка стилей в DevTools.	2	1	1
19-20	Цвета, единицы измерения и типографика	Подбор палитры, работа с rem/em/vw. Подключение веб-шрифтов и настройка читабельности текста.	2	1	1
21-22	Flexbox: выравнивание и раскладка	Изучение модели flexbox. Создание гибких рядов и колонок на flexbox. Выравнивание элементов и сборка сетки карточек.	2	1	1
23-24	Паттерны Flex и подход mobile-first	Изучение верстки навигации и медиа-блоков. Учимся сначала делать мобильную версию, затем расширять под планшет и десктоп. Написание структуры сайта с использованием flexbox.	2	1	1
25-26	CSS Grid: основы	Освоение CSS Grid: треки, фракции и авторазмещение элементов. Построение простой сетки.	2	1	1
27-28	CSS Grid: области и адаптивная сетка	Изучение областей в модели Grid. Создание области макета (grid-template-areas) и адаптивных сеток с minmax и auto-fit/fill.	2	1	1
29-30	Адаптивность: медиазапросы и «резиновая» типографика	Освоение методов написания адаптивных страниц. Адаптивность сайта: настройка @media и функции clamp() для «резиновой» типографики. Обсуждение контейнерных запросов.	2	1	1
31-32	Адаптивные изображения	Объяснение необходимости оптимизации в веб-приложениях. Оптимизация изображений: использование srcset/sizes, тега picture и формата WebP/SVG.	2	1	1
33-36	Практика по модулю 3	Выполнение практического задания по модулю 3.	4	-	4

Модуль 4: Работа с Figma			**6**	**2**	**4**
37-38	Figma для разработчика	Учимся извлекать дизайн из Figma: кадры, авто-layout, сетки и экспорт графики для веб-сайта. Отработка знаний на практике.	2	1	1
39-40	Перенос макетов из Figma	Знакомство с дизайн-токенами и CSS-переменными. Пошаговый перенос макета из Figma в код.	2	1	1
41-42	Практика по модулю 4	Выполнение практического задания по модулю 4.	2	-	2
Модуль 5: Работа с Git и Github			**6**	**3**	**3**
43-44	Основы Git: сохранение истории проекта	Система контроля версий git. Изучение работы с сайтом gitlab. Инициализация репозитория, отправка коммитов, настройка .gitignore и работы по SSH.	2	1	1
45-46	Командная работа: ветки, Pull Request и ревью	Объяснение необходимости проверки кода в команде. Изучение понятий pull-request (PR) и ревью. Работа в команде: создание ветки, открытие Pull Request, проведение ревью и объединение изменений.	2	1	1
47-48	Публикация сайта (GitHubPages/Vercel)	Рассмотрение методов развертывания сайта на сервере. Публикация сайта по инструкции на GitHubPages/Vercel.	2	1	1
Модуль 6: Введение в программирование на JavaScript			**18**	**8**	**10**
49-50	JavaScript: базовый синтаксис и типы	Изучение переменных, операций и преобразований типов. Подготовка к решению задач.	2	1	1
51-52	JavaScript: условные конструкции	Изучение понятий: условие, условная конструкция. Элементы if, else if, else. Написание условных конструкций в JavaScript. Решение нескольких задач разной сложности.	2	1	1
53-54	JavaScript: циклы	Понятия цикл, счетчик. Основные виды циклов: for, while, do while. Повторение блоков кода несколько раз. Написание циклов в JavaScript. Решение нескольких задач разной сложности.	2	1	1

55-56	Функции и область видимости	Понятия функция, аргумент, параметр. Определение и вызов функций, использование стрелочного синтаксиса. Понимание идеи замыканий.	2	1	1
57-58	Массивы и объекты в JavaScript	Изучение коллекций в JavaScript. Понятия массив и объект. Работа с массивами и объектами: map, filter, reduce, деструктуризация.	2	1	1
59-60	Модули ES: структура и переиспользование	Учимся разбивать код на модули ES: импорт/экспорт, структура файлов и «чистые функции». Разбиение кода на модули при решении задач.	2	1	1
61-62	Отладка кода и обработка ошибок	Знакомство с обработкой ошибок. Отладка кода через DevTools, ставим точки останова и обрабатываем ошибки с try/catch.	2	1	1
63-64	Рефакторинг и защита от ошибок	Изучение продвинутого использования функций: добавление проверки и значения по умолчанию, повышение читаемости и надёжности. Проводим рефакторинг кода и добавляем защиту от ошибок.	2	1	1
Промежуточная аттестация					
65-66	**Промежуточная аттестация**	**Выполнение задания за ограниченное время**	2	-	2
Модуль 7: DOM и интерактивность веб-страниц			**22**	**9**	**13**
67-68	DOM: поиск и изменение элементов	Освоение поиска и изменения элементов на странице: текст, атрибуты и классы. Переключение тёмной/светлой темы. Решение практических задач на поиск и изменение элементов.	2	1	1
69-70	События в браузере и делегирование	Понимание жизненного цикла событий в браузере. Добавление обработчиков, использование всплытия и делегирования для списков и галерей.	2	1	1
71-72	Формы: клиентская проверка данных	Как создавать формы на стороне клиента. Проверка данных на стороне клиента: отображение ошибок у полей и учет доступности.	2	1	1
73-74	Сохранение данных в браузере	Объяснение возможностей сохранения данных в браузере. Сохранение данных в браузере через localStorage и JSON: настройки, черновики и тема.	2	1	1

75-76	Работа с API: запросы и ответы	Изучение сетевых запросов. Отправка запросов к API с fetch и async/await. Обработка статусов и сетевых ошибок.	2	1	1
77-78	Пользовательские состояния интерфейса	Пояснение UI/UX со стороны пользователя. Добавление состояний интерфейса: загрузка, пусто, ошибка, успех. Создание кнопки «повторить».	2	1	1
79-80	Подгрузка данных: пагинация и бесконечный скролл	Что такое динамическая загрузка данных. Понятия пагинация и бесконечный скролл. Реализация подгрузки карточек: параметры запросов, кнопка «еще» и бесконечный скролл через IntersectionObserver.	2	1	1
81-82	Поиск и фильтрация данных	Изучение возможностей реализации поиска и фильтрации. Создаём поиск с debounce, фильтры и сохраняем выбор пользователя.	2	1	1
83-84	Подключение внешних API	Чтение документации по API, учитывая лимиты и ключи. При необходимости использование локального json-server. Подключение внешнего API к сайту.	2	1	1
85-88	Практика по модулю 7	Выполнение практического задания по итогам освоения модуля 7.	4	-	4
Модуль 8: Сборка и форматирование проектов			**6**	**3**	**3**
89-90	Сборка проекта: npm-скрипты, Vite и ENV	Что такое сборщики и для чего они нужны. Настройка сборки проекта, алиасы путей и команды dev/build/preview. Работа с ENV.	2	1	1
91-92	Единый стиль кода: ESLint и Prettier	Как привести проект к единому стилю. Выравнивание стиля кода: подключение ESLint и Prettier.	2	1	1
93-94	Как устроен проект: модули и компоненты; подключение библиотек	Организация проекта по функциональным модулям. Сборка переиспользуемых UI-блоков. Популярные архитектуры проектов. Установка библиотек в проекты.	2	1	1
Модуль 9: React и TypeScript			**22**	**9**	**13**
95-96	React: первый компонент	Знакомство с компонентным подходом при создании сайтов. Создание проекта на Vite+React, написание первого компонента на JSX и передача данных через props.	2	1	1

97-98	React: состояние и события	Понятия реактивности и обновления данных в реальном времени через состояния. Добавление состояния через useState, обработка события. Примеры: счётчик, лайки, фильтр.	2	1	1
99-100	React: эффекты и загрузка данных	Понятие жизненного цикла компонента и динамической подгрузки данных. Подгрузка данных в useEffect и очистка эффектов при размонтировании компонента. Жизненный цикл компонента.	2	1	1
101-102	Введение в TypeScript	Что такое TypeScript и чем он отличается от JavaScript. Основные типы: string, number, boolean, any. Типизация переменных и функций в проекте.	2	1	1
103-104	TypeScript в React-компонентах	Изучение типизации передаваемых значений, функций и состояний. Типизация props через interface и type. Типизация состояния: useState<number>, useState<string>. Типизация обработчиков событий: MouseEvent, ChangeEvent.	2	1	1
105-106	ReactRouter: навигация между страницами	Понятия одностраничных сайтов (SPA) и многостраничных (MPA). Как настроить маршрутизацию для создания эффекта страниц. Добавление маршрутов и ссылок, чтение параметров из адресной строки.	2	1	1
107-108	React: формы и контроль ввода	Что такое управляемые формы и как их создавать. Создание управляемых полей, вывод ошибок и отправка данных формы.	2	1	1
109-110	React: списки и ключи элементов	Как отрисовываются элементы из списков. Обязательный параметр key при отрисовке из списков. Отрисовка списков и установка корректных ключей, чтобы избежать лишних перерисовок.	2	1	1
111-112	React: простые кастомные хуки	Что такое кастомный хук. Вынос общей логики в простые хуки (useFetch/useDebounce) и упрощение компонентов. Создание кастомных хуков на практике.	2	1	1
113-116	Практика по модулю 9	Выполнение практического задания по итогам освоения модуля 9.	4	-	4
Модуль 10: Итоговый проект			**20**	**-**	**20**
117-118	Итоговый проект: постановка задачи	Формулировка ТЗ, определение пользователей и их задач. Согласование критериев успеха.	2	-	2
119-120	Итоговый проект: структура и контент	Построение карты сайта и модели контента. Подготовка текстов и изображений.	2	-	2

121-122	Проектный спринт I: каркас и сетки	Верстка базового каркаса, сетки и дизайн-токенов. Подготовка skeleton-экранов.	2	-	2
123-124	Проектный спринт I: адаптив и изображения	Добавление брейк поинтов и адаптивных изображений. Проверка сайта в Lighthouse.	2	-	2
125-126	Проектный спринт I: модули и переиспользование	Выделение UI-компонентов и подключение системных переменных CSS.	2	-	2
127-128	Проектный спринт II: навигация и формы	Добавление маршрутизации (или навигации) и проверка форм.	2	-	2
129-130	Проектный спринт II: данные и API	Интеграция API или моковых данных. Отображение списков и страниц деталей.	2	-	2
131-132	Проектный спринт II: фильтры и поиск	Реализация поиска с debounce и сохранение выбранных фильтров.	2	-	2
133-134	Финальная подготовка к защите	Фикс багов, деплой проекта. Подготовка к презентации.	2	-	2
Итоговая аттестация					
135-136	**Итоговая аттестация.**	**Защита и презентация проекта, ответы на вопросы комиссии.**	**2**	**-**	**2**
Итого:			**136**	**48**	**88**

1.3.2 Содержание учебного плана

Модуль 1. Введение в веб-разработку.

1.1. Тема: Старт курса и установка инструментов.

Теория: Знакомство, командообразование, инструктаж по ТБ и поведению в учебном кабинете. Правила работы с компьютером. Входная диагностика в форме опроса. Цифровая гигиена. Как работает веб: клиент ↔ сервер, URL, запрос/ответ (HTTP), роль браузера и DevTools. Обзор HTML/CSS/JS и что мы будем уметь к концу курса.

Практика: Установка Node.js, Git, VS Code; настройка базовых расширений. Создание папки проекта, «Hello, Web» в браузере, первое открытие DevTools.

1.2. Тема: Первый проект.

Теория: Что такое репозиторий, коммит, gitignore. Структура HTML-страницы (doctype, head, body), где подключать CSS и JS.

Практика: gitinit и первый коммит с README.md; создание index.html, подключение style.css и main.js; базовая разметка страницы.

Модуль 2. Основы разметки веб-страниц

2.1. Тема: Семантическая разметка HTML5: смысловые теги контента.

Теория: Разбираем, как HTML передаёт смысл, а не внешний вид: заголовки формируют «оглавление» страницы и помогают пользователю (и поисковику) понять структуру. Ссылки задают связи и маршруты, изображения — смысловой контент, который должен быть понятен и без картинки (через вдумчивый alt). Понимаем, почему «наставить div'ов» — это хуже для доступности, SEO и поддержки, чем использовать корректные смысловые элементы.

Практика: Превращаем «плоский» текстовый набросок в структурированную статью: наводим иерархию заголовков, выделяем смысловые блоки, оформляем ссылки «говорящими» текстами, осмысленно подписываем

изображения через подписи/альтернативы. Критерий готовности: по одному взгляду на оглавление понятно, где что искать.

2.2. Тема: Семантическая структура страницы: области.

Теория: Учимся строить «скелет» страницы — landmarks: шапка, навигация, основной контент, боковые вставки и подвал. Это то, чем пользуются скринридеры и клавиатурные пользователи для быстрой навигации. Обсуждаем правило одного main на страницу, как помечать несколько навигаций и зачем нужен skip-link («перейти к содержимому»).

Практика: Проектируем каркас лендинга: выделяем области, делаем логичную навигацию по якорям, проверяем, что клавиатурой можно быстро перейти к контенту. Критерий готовности: карту страницы (шапка/навигация/контент/подвал) можно «прочитать» с клавиатуры без мыши.

2.3. Тема: Медиа-контент и подписи.

Теория: Картинка в вебе — это данные, а не «украшение». Учимся различать, когда у изображения нужен описательный alt, а когда он должен быть пустым (декор). Подписи под медиа помогают связать изображение с текстом и контекстом. Коротко обсуждаем форматы и сетку/кадрирование: не растягивать, не «ломать» пропорции, давать смысл.

Практика: Собираем мини-галерею с осмысленными подписями и корректными альтернативными описаниями для каждого случая (инфографика, фото, декоративная иконка). Критерий готовности: если отключить картинки, смысл блока сохраняется.

2.4. Тема: Таблицы данных в HTML.

Теория: Рассматриваем таблицу как структуру отношений: заголовки столбцов/строк задают смысл ячеек, а не просто «сетка». Понимаем роль thead/tbody/tfoot, ассоциацию заголовка с ячейкой (scope), зачем нужен caption. Разбираем, почему таблица не подходит для верстки макета, а годится именно для данных.

Практика: Проектируем таблицу расписания/прайса: формулируем заголовки так, чтобы без визуала было понятно, «что за что отвечает», отмеча-

ем тоталы в tfoot. Добавляем минимальную стилизацию на читабельность (зебра/выравнивание). Критерий готовности: таблицу можно «прочитать» линейно и понять связи.

2.5. Тема: Формы: поля ввода и доступность.

Теория: Смотрим на форму как на сценарий: пользователь приходит с задачей, интерфейс должен его направлять. Связанные label и поля повышают понятность и доступность; правильные типы (email, tel, number) дают валидацию «из коробки» и удобные клавиатуры на мобильных. Обсуждаем обязательность, подсказки и микро-копирайтинг ошибок так, чтобы пользователь понимал, что исправлять.

Практика: Делаем форму обратной связи: короткие и понятные подписи, required только там, где нужно, осмысленные сообщения ошибок, фокус по Tab, кнопка отправки в конце логического потока. Финальные 15–20 минут — мини-итог модуля: быстрая самопроверка страницы (оглавление, landmarks, галерея, таблица, форма) по чек-листу доступности. Критерий готовности: форму можно заполнить без мыши, ошибки понятны, введённые данные не теряются при повторной попытке.

2.6. Тема: Практика по итогам освоения модуля 2.

Практика: Собрать одну полноценную HTML-страницу “Инфораздел” (например, «Курс/События/Контакты») без JS, с минимальными стилями или вовсе без CSS — фокус на разметке и доступности.

Модуль 3: Оформление и макетирование страниц.

3.1. Тема: Основы CSS: каскад и блочная модель.

Теория: Разбираем, как браузер решает, какие стили применить: источник стилей (браузерные, пользовательские, авторские), наследование, специфичность и порядок следования — это и есть «каскад». Понимаем box-model: контент, padding, border, margin, влияние box-sizing. Смотрим, как элементы участвуют в потоке (display: block/inline/inline-block) и почему «слетает» верстка при забытом отступе. Учимся отлаживать стили в

DevTools: смотреть применённые правила, считать итоговые размеры, ловить конфликтующие селекторы.

Практика: Верстаём карточку контента (заголовок, текст, кнопка): настраиваем внутренние отступы так, чтобы кликабельная область кнопки была комфортной; делаем визуально ровные поля между элементами. Упражнение «найди и исправь»: ловим проблемы с «прыгающими» отступами и неочевидной специфичностью (находим и обосновываем правки).

3.2. Тема: Цвета, единицы измерения и типографика.

Теория: Строим систему размеров: абсолютные (px) vs относительные (rem, em, vw/vh) — почему лучше масштабировать от базового размера шрифта. Вводим типографику: кегль, межстрочный интервал, интервалы между параграфами и заголовками; читаемость на разных экранах. Формируем цветовую палитру (акцентный, фоновые, текстовые) и обсуждаем контраст (доступность). Коротко о подключении веб-шрифтов и влиянии на производительность.

Практика: Задаём типографическую шкалу (например, через rem + clamp()), создаём CSS-переменные для цветов и размеров. Приводим 2–3 страницы к единому «ритму»: заголовки выглядят согласованно, текст читается, контраст выдержан. Чек-лист: заголовки не «прыгают», интерлиньяж комфортный, акцентный цвет читаем поверх фона.

3.3. Тема: Flexbox: выравнивание и раскладка.

Теория: Flex — это одномерная раскладка: контейнер, главная/перекрёстная оси, justify-content/align-items, управление шириной через flex: growshrinkbasis. Обсуждаем «подводные камни»: почему элементы иногда не ужимаются (min-width: 0), как работает gap, когда лучше не тянуть все блоки равномерно. Показываем типичные шаблоны: «шапка» с логотипом, меню и действиями справа; ряд карточек с переносом.

Практика: Собираем хедер: логотип слева, навигация по центру, кнопка/аватар справа; проверяем поведение при сужении экрана. Делаем сетку карточек на flex с равномерными промежутками и выравниванием по вы-

соте. Контрольный тест: быстро перестроить порядок/выравнивание без изменения HTML — только стилями.

3.4. Тема: Паттерны Flex и подход mobile-first.

Теория: Применяемflex к реальным шаблонам: media-object (картинка + текст), «две колонки с акцентом», «полоска действий». Объясняем mobile-first: пишем стили для узкого экрана как базу, затем расширяем (минимальные брейкпоинты). Почему так устойчивее: меньше переопределений, лучше производительность. Коротко про размеры кликабельных зон и отступы для тач-устройств.

Практика: Рефакторим секцию «О нас» под mobile-first: сначала одна колонка со смысловым порядком, затем на широких экранах — две/три колонки. Добавляем брейкпоинт(ы) и докручиваем выравнивание/промежутки. Проверяем в DevTools (Device Toolbar) несколько пресетов экранов. Критерий готовности: контент читаем и удобен без горизонтальной прокрутки, элементы не «липнут» к краям, акценты сохранены.

3.5. Тема: CSS Grid: основы.

Теория: Сравниваем Grid и Flex: Grid — про две оси сразу (строки и колонки), удобно для сложных макетов. Разбираем понятия явной/неявной сетки, треки (строки/колонки), единицу fr, зазоры gap, автоматическое размещение (auto-placement). Обсуждаем выбор размеров: фиксированные (px), гибкие (fr), содержимое-зависимые (auto). Показываем типичные ошибки: «ломка» сетки из-за больших картинок, отсутствие ограничений по ширине, смешение отступов и gap.

Практика: Делаем галерею/портфолио: сетка 3×N на десктопе с аккуратным gap, карточки одинаковой ширины, подпись под каждой. Проверяем поведение при узкой ширине экрана: как минимум — плавное сжатие, без горизонтальной прокрутки. Бонус: аккуратно ограничить рост изображений (max-width: 100%, object-fit/aspect-ratio по необходимости).

3.6. Тема: CSS Grid: области и адаптивная сетка.

Теория: Вводимgrid-template-areas: читаемо задаём «каркас страницы» (шапка, навигация, контент, сайдбар, подвал). Используем minmax() для разумного «дыхания» колонок. Демонстрируем auto-fit/auto-fill: как автоматически заполнять пространство карточками без медиазапросов. Паттерн «сайт-каркас»: одна колонка на мобильном → две/три на широких экранах, без дублирования HTML.

Практика: Собираемлэйаут страницы: header сверху на всю ширину, main + aside в середине, footer снизу. На мобильном — один столбец; на планшете — main шире, aside уже; на десктопе — устойчивые поля и равномерные межблочные расстояния. Проверяем читаемость и порядок фокуса.

3.7. Тема: Адаптивность: медиазапросы и «резиновая» типографика.

Теория: Переходим от «разрешение-ориентированных» брейкпоинтов к контент-ориентированным: ставим точку перелома там, где ломается композиция. Стратегия mobile-first: базовые стили для узких экранов, далее расширяем через @media (min-width: …). Вводим флюидную типографику через clamp() — чтобы заголовки и базовый шрифт масштабировались мягко между min/max, а не «скакали» по ступенькам. Кратко затрагиваем контейнерные запросы как перспективу.

Практика: Задаём 2–3 брейкпоинта для уже верстаемых секций. Переписываем типографику на clamp() (например, базовый font-size, размеры h1–h3). Тестируем в DevTools разные девайсы и масштаб, проверяем, что строки не становятся слишком длинными/короткими и сохраняется удобочитаемость.

3.8. Тема: Адаптивные изображения.

Теория: Задача картинок — быть чёткими и лёгкими на любом экране. Разбираем srcset/sizes для выбора подходящего размера под плотность и ширину, picture для арт-направления (другая обрезка на мобайле) и форматы (WebP/SVG; упоминаем AVIF как опцию). Профилактика визуальных скач-

ков (CLS): заранее задаём width/height или используем aspect-ratio; отложенная загрузка loading="lazy" для невидимых медиа.

Практика: Переделываемhero-блок и галерею: добавляем srcset/sizes, при необходимости picture с альтернативами форматов, включаем loading="lazy" у карточек ниже первого экрана. Сверяем в Lighthouse/Performance: нет ли «скачков» макета при загрузке, не перегружаем ли мобильную сеть.

3.9. Тема: Практика по итогам освоения модуля 3.

Практика: Собираем 2–3 готовые секции в проекте (hero + features + testimonials/cta), прогоняем мини-аудит (Lighthouse + ручной a11y-чек): фиксируем 2–3 найденных проблемы и тут же устраняем. Итог — «полка» устойчивых адаптивных блоков, готовых для дальнейшей интеграции.

Модуль 4: Работа с Figma.

4.1. Тема: Figma для разработчика.

Теория: Учимся смотреть на макет как разработчик: где границы экранов, сетка (layoutgrid), отступы и Auto Layout как аналог flex-раскладок. Разбираем constraints (прилипание к краям) — почему при сужении экрана кнопка остаётся на месте. Обсуждаем FigmaVariables/стили (цвет, текст, эффекты) и как превратить их в CSS-переменные. Экспорт графики: когда SVG (иконки, логотипы), когда WebP/PNG, плотности @1x/@2x. Про авторские права: откуда брать изображения и как указывать источники.

Практика: В инспекторе Figma фиксируем реальные значения (пэддинги, gap, размеры шрифтов) и пересобираем секцию «hero» в HTML/CSS. Выгружаем 2–3 иконки в SVG, один герой-баннер в WebP в нужном размере. Создаём стартовую таблицу дизайн-токенов (цвета/шрифты/spacing) для будущих CSS-переменных.

4.2. Тема: Перенос макетов из Figma.

Теория: Типичные расхождения макет↔код: межстрочные интервалы, невидимые пэддинги контейнеров, неправильные шрифты/веса, забытые gap.

Как оценивать «достаточно близко», чтобы не тратить часы на выравнивание в пиксель.

Практика: Каждый выбирает 1–2 секции из макета (hero/преимущества/карточки) и переносит в код. Критерии сдачи: совпадают размеры иерархии шрифтов, поля между блоками, сетка «держит» при 2–3 ширинах экрана; изображения экспортированы корректно; базовая доступность сохранена (правильные заголовки, читаемый контраст).

4.3. Тема: Практика по итогам освоения модуля 4.

Практика: Переписываем существующие стили в проекте на переменные (:root{ --color-primary… }), подгоняем 2–3 секции под единую шкалу. Делаем короткий «регресс-чек»: ничего не «поехало», токены действительно переиспользуются, а не дублируются.

Модуль 5: Работа с Git и Github.

5.1. Тема: Основы Git: сохранение истории проекта.

Теория: Git как «машина времени»: атомарные коммиты, говорящие сообщения (feat:,fix:,refactor:), зачем .gitignore. Настройка имени/почты, SSH-ключи. Базовые операции: status/add/commit/log/diff, возврат к версии, «safereset». Почему частые маленькие коммиты лучше редких больших.

Практика: Инициализация репозитория проекта, настройка .gitignore. Серия осмысленных коммитов по чек-листу (добавили секцию, отвязали стили, поправили токены). Учимся смотреть gitlog --oneline --graph, делать gitdiff и понимать, что именно изменили.

5.2. Тема: Командная работа: ветки, Pull Request и ревью.

Теория: GitHubFlow: main стабилен, фичи — в ветках. PR как место обсуждения. Как писать чек-лист ревью (читабельность, доступность, адаптивность, производительность). Конфликты слияния — почему возникают и как их предупредить.

Практика: Каждый создает ветку feature/…, делает изменение и открывает Pull Request. Проводим взаимное ревью: комментируем строки, про-

сим правки, добавляем approve. Закрываем PR (merge/squash) и проверяем, что история чистая.

5.3. Тема: Публикация сайта (GitHubPages/Vercel).

Теория: Что такое деплой и в чём разница: статическая страница (GitHubPages) и SPA со сборкой (Vercel/Netlify). Базовые понятия: билд, корень проекта, «fallback» для роутера SPA, переменные окружения (ENV) и почему секреты нельзя класть в клиент. Мини-ввод про домены/URL.

Практика: Публикуем учебный сайт:

- Вариант А (статический): GitHubPages → проверяем, что стили/картинки подхватились, ссылки работают.
- Вариант Б (SPA на Vite): Vercel → собираем, настраиваем SPA fallback (иначе 404 при прямом заходе на /route).

Фиксируем живую ссылку, прогоняем Lighthouse, проверяем метрики и доступность.

Модуль 6: Введение в программирование на JavaScript.

6.1. Тема: JavaScript: базовый синтаксис и типы.

Теория: Разбираем, как JS хранит данные и выполняет инструкции: переменные let/const (почему избегаем var), примитивные типы (число, строка, булево, null, undefined) и преобразования между ними в реальной жизни (ввод из формы всегда строка). Учимся составлять выражения, использовать операторы, шаблонные строки для аккуратного текста, и понимать, где возникают типичные ловушки (сложение строк и чисел).

Практика: Пишем маленькие «утилиты»: калькулятор стоимости заказа (цена × количество + доставка), форматирование приветствия по введённому имени, конвертер единиц. Цель — аккуратно получать данные из переменных, приводить типы и выводить результат в консоль.

6.2. Тема: JavaScript: условные конструкции.

Теория: Делаем код «решающим»: ветвление с if/else и switch для разных сценариев, логические операторы для компактных проверок.

Практика: Серия задач «жизненных правил»: определение скидки по возрасту, подбор сообщения об ошибке по коду.

6.3. Тема: JavaScript: циклы.

Теория: Повторяем действия с for и while, обсуждаем, как выбирать границы и условия, чтобы не зациклиться. Паттерн «сторожевой» проверки входных данных перед вычислениями.

Практика: Серия задач «жизненных правил»: генерация таблицы (например, расписания) циклом. Критерий: ни одна ветка не остаётся «висячей», для каждого входа есть предсказуемый выход.

6.4. Тема: Функции и область видимости.

Теория: Функция как способ дать имени шагам решения: параметры, возвращаемые значения, чистые функции и побочные эффекты (почему полезно разделять). Стрелочные функции как короткая форма. Область видимости: локальные переменные не «утекают» наружу; где хранить состояние, чтобы не мешать другим частям кода.

Практика: Разбиваем «толстую» задачу на функции: расчёт суммы, форматирование текста, рендер карточки. Пишем мини-«тесты» в консоли (несколько вызовов с разными аргументами) и фиксируем ошибки.

6.5. Тема: Массивы и объекты в JavaScript.

Теория: Массив как упорядоченная коллекция, объект как словарь свойств. Учимся доставать/менять элементы, добавлять/удалять. Современные инструменты: деструктуризация, спред, итеративные методы (map, filter, reduce) как способ выразить «что хотим получить», а не «как это пробежать».

Практика: Делаем мини-каталог «товаров»: фильтруем по цене и категории, сортируем по названию, считаем суммарную стоимость через reduce. Рендерим предварительно в консоль, готовим данные для будущего UI.

6.6. Тема: Модули ES: структура и переиспользование.

Теория: Почему «всё в одном файле» плохо: повтор, сложности поиска и поддержки. Модули ES (import/export) как способ разделить ответственность: «данные», «логика», «отрисовка». Обсуждаем относительные пути и

нейминг (понятные имена файлов/функций), принцип «один модуль — одна цель».

Практика: Раскладываем код мини-каталога на модули: data.js (исходные данные), filters.js (операции с массивом), render.js (текстовыйрендер). Подключаем их в main.js и проверяем, что логика изолирована и переиспользуема.

6.7. Тема: Отладка кода и обработка ошибок.

Теория: Учимся «видеть» код в движении: DevTools → вкладка Sources, точки останова, пошаговое выполнение, «watch» за переменными, стек вызовов. Логирование как инструмент (но не костыль). Обработка ошибок try/catch: где ловить и что показывать пользователю (не «сломанные» сообщения вроде [object Object]). Разница между ошибками данных и ошибками программиста.

Практика: Получаем «сломанный» пример (намеренные баги: опечатки, неверные типы, ошибка в условии). Ставим брейкпоинты, находим место поломки, добавляем try/catch вокруг опасной операции (например, парсинг JSON или вычисление с пустыми данными), выводим дружелюбное сообщение и защищаем функции «сторожевыми» проверками. Результат — чек-лист найденных багов и краткое описание исправления.

6.8. Тема: Рефакторинг и защита от ошибок.

Теория: Разбираем приёмы защитного программирования: предварительные проверки входных данных (гварды), значения по умолчанию, ранний выход из функции. Обсуждаем «запахи кода»: дублирование, чрезмерно длинные функции, «магические числа». Показываю, как осмысленные имена, разбиение на маленькие функции и перенос повторяющейся логики в утилиты снижают количество багов. Касаемся иммутабельности (не менять входные данные) и «контрактов» функций (что принимают, что возвращают).

Практика: Берём ранее написанную мини-фичу (например, каталог с фильтрами) и делаем рефакторинг: выносим повторы в helpers, добавляем гварды, убираем «магические» значения в константы, пишем 3–4 «самотеста»

в консоли с разными входами (в т.ч. пустыми/не валидными). Фиксируем до/после: стало ли легче читать и исправлять.

6.9. Тема: Промежуточная аттестация.

Практика: За ограниченное время делаем мини-страницу-виджет:

- Адаптивная секция (герой/карточки) с корректной типографикой.
- Форма с валидацией (минимум 3 поля, дружелюбные сообщения).
- Маленькая интерактивность на событиях (например, фильтр списка или раскрывающийся блок) с сохранением одного параметра в localStorage.

Финал: Рефлексия, короткий самоотчёт — что получилось, где пришлось импровизировать, один баг, который нашли и исправили.

Модуль 7: DOM и интерактивность веб-страниц.

7.1. Тема: DOM: поиск и изменение элементов.

Теория: DOM как дерево узлов: как находить нужный элемент, когда менять textContent, а когда безопасно (и когда опасно) вставлять HTML. Обсуждаем управление классами vsинлайн-стилями, как минимизировать «мигание» интерфейса (группировать изменения). Касаемся доступности: при динамической подмене текста важно сохранять смысл и не ломать порядок фокуса.

Практика: Реализуем переключатель темы (светлая/тёмная) и раскрывающиеся блоки («подробности/свернуть»). Тренируемся менять атрибуты, добавлять/удалять классы, аккуратно обновлять текст. Критерий: без перезагрузки страницы контент меняется предсказуемо, ничего не «скачет» и остаётся доступным с клавиатуры.

7.2. Тема: События в браузере и делегирование.

Теория: Понимаем путь события (захват → цель → всплытие), различие event.target и currentTarget, когда уместно preventDefault и stopPropagation. Почему делегирование на контейнер экономит память и упрощает код для списков, меню и галерей. Затрагиваем клавиатурные сценарии: Enter/Space для кнопок, Escape для закрытия.

Практика: Делаем интерактивный список/галерею с одним обработчиком на контейнер: добавление/удаление элементов, отметка «избранное», просмотр подробностей по клику/Enter. Проверяем, что клики по вложенным элементам корректно обрабатываются и код остаётся коротким.

7.3. Тема: Формы: клиентская проверка данных.

Теория: Используем то, что даёт браузер (ConstraintValidation API) и добавляем свою логику для понятных сообщений. Обсуждаем, где проверять — «на лету» или при отправке, — чтобы не раздражать пользователя. Показываю паттерн aria-live для озвучивания ошибок скринридером и фокус-менеджмент: перенос фокуса к первому проблемному полю. Кратко про базовую санацию строк (обрезка пробелов, ограничение длины).

Практика: Строим форму (имя, email, сообщение) с живой валидацией: подсветка поля, краткое пояснение рядом, блокировка отправки до исправления. На «Отправить» — предотвращаем дефолт, если есть ошибки; если всё ок — показываем «данные приняты» и очищаем форму. Цель — комфортная и доступная UX-валидация.

7.4. Тема: Сохранение данных в браузере.

Теория: Для чего нужны localStorage и sessionStorage, как сериализовать данные в JSON, ограничения по объёму и приватности (секреты не хранить). Паттерны: сохранение настроек интерфейса (тема, язык), черновики форм (чтобы не терялись при перезагрузке), простое кеширование результатов запроса. Разбираем, как корректно инициализировать состояние при загрузке страницы.

Практика: Сохраняем выбор темы из прошлых занятий, добавляем автосохранение черновика формы из 63–64 и делаем «избранное» для карточек каталога (список хранится в localStorage). Проверяем: после перезагрузки всё восстанавливается; есть очистка и понятная индикация сохранённого состояния.

7.5. Тема: Работа с API: запросы и ответы.

Теория: Что такое сеть «с точки зрения JS»: запрос уходит, ответ приходит позже — поэтому нужны промисы и async/await. Разбираем жизненный цикл запроса: fetch → Response (status, headers) → .json(); отличие сетевой ошибки от «успешного ответа со статусом 404/500». Говорим про CORS, таймауты (через AbortController), повтор запроса и «чистый» интерфейс функции загрузки (одна точка входа, проверка статуса, парсинг JSON, выбрасывание осмысленных ошибок).

Практика: Пишем обёртку request(url, options): проверяет статус, парсит JSON, умеет отменять запрос. Собираем небольшой список карточек из публичного API (или подготовленного мок-JSON), отображаем основные поля. Делаем первый минимальный обработчик ошибки: если что-то пошло не так — показываем понятное сообщение и даём ссылку «повторить».

7.6. Тема: Пользовательские состояния интерфейса.

Теория: Пользователь должен понимать «что происходит»: состояния loading / empty / error / success; когда и чем они отличаются. Анти-паттерны: «бесконечный спиннер» без пояснений, исчезающие ошибки. Обсуждаемretry, отключение кнопок на время запроса, «идемпотентные» повторные клики. Учитываем нестабильную сеть, медленные ответы и офлайн (Navigator.onLine — как намёк).

Практика: Доделываем список для выбранной темы:

- «скелетон» или спиннер на время загрузки,
- экран empty (нет результатов) с советом,
- аккуратный error с кнопкой «повторить»,
- нормальный success.

Проверяем руками сценарии: долго грузится, сервер вернул 204/пустой массив, ошибка сети, быстрый успех.

7.7. Тема: Подгрузка данных: пагинация и бесконечный скролл.

Теория: Два подхода на сервере: page/limit и cursor/next.Почемуcursor устойчивее к изменениям данных. Клиентские задачи: не дублировать элементы, сохранять позицию скролла, уметь отменять «устаревший» запрос

при повторных кликах. Показываем IntersectionObserver для «ленивой» подгрузки и аккуратно говорим про виртуализацию списков (как перспектива).

Практика: Делаем «Показать ещё» с page/limit, потом апгрейд до infinitescroll через «сторожок» (sentinel). Обязательно: disabled-состояние кнопки во время загрузки, защита от двойных кликов, отмена предыдущего запроса при смене источника данных.

7.8. Тема: Поиск и фильтрация данных.

Теория: Debounce поля ввода: не слать запрос на каждую клавишу, выбирать задержку 250–400 мс. Разница клиентской фильтрации (по уже загруженным данным) и серверной (с параметрами в URL). Полезно хранить критерии в URL-строке (queryparams) и/или localStorage, чтобы состояние не терялось при обновлении страницы. Важный UX: «ничего не найдено» — это нормальный результат, а не молчание.

Практика: Добавляем строку поиска и 1–2 фильтра (например, категория/диапазон). Реализуем debounce и обновление списка; выбранные фильтры пишем в URL и восстанавливаем при загрузке. Проверяем совместимость с пагинацией: при смене фильтра сбрасываем на первую страницу, отменяем «старый» запрос.

7.9. Тема: Подключение внешних API.

Теория: Что такое API-ключ и почему его нельзя «прятать» в клиентском коде. В учебных проектах: работаем с открытыми эндпоинтами, моками или прокси; вводим ограничители: лимит попыток, экспоненциальный бэкофф, кеширование результата в памяти на сессию. Осмысленные сообщения пользователю: различаем «лимит исчерпан» и «временная недоступность». Коротко про .env для сборщика и про мок-серверы.

Практика: Подключаем второй эндпоинт (или подменяем реальный — на json-server/mock). Встраиваем бэкофф/кеш в нашу обёртку request. Делаем «демо-переключатель»: реальный API ↔ мок-данные, чтобы приложение было проверяемым на занятии даже без сети/ключа.

7.10. Тема: Проверочная работа по итогам освоения модуля 7.

Практика: Работаем командами: соединяем поиск/фильтры/пагинацию в один модуль, устраняем дублирование, улучшаем метрики (убираем лишние перерисовки, включаем loading="lazy" для изображений, сокращаем количество запросов). Финальный PR с ревью по чеклисту: что почистили, что ускорили, какие кейсы покрыли.

Модуль 8: Сборка и форматирование проектов.

8.1. Тема: Сборка проекта: npm-скрипты, Vite и ENV.

Теория: Разбираем экосистему: npm как менеджер пакетов и сценариев, почему сборщик (Vite) ускоряет разработку (горячая перезагрузка, минификация, импорт модулей). Что такое переменные окружения (.env) и зачем выносить туда URL API и «флаги» (но не секреты). Задача алиасов путей — побороть «../../../../component», сделать импорты короткими и читаемыми. Обсуждаем различия dev/build/preview и как проверить, что «в проде» всё работает.

Практика: Инициализируем проект на Vite, добавляем npm-скрипты (dev, build, preview). Создаём .env с VITE_API_BASE, настраиваем алиасы (@/components, @/lib) и переносим 1–2 импорта на алиасы. Сборка + локальный предпросмотр: проверяем, что приложение поднимается и обращается к правильному базовому URL.

8.2. Тема: Единый стиль кода: ESLint и Prettier.

Теория: Зачем автоматизировать стиль и качество: меньше спорных ревью, быстрее читаем код. ESLint ловит ошибки/ «запахи», Prettier — форматирует однозначно. Договариваемся о базовых правилах (имена, кавычки, запятые, пробелы, порядок импортов), обсуждаем «сигналы» линтера: где важно прислушаться, а где осознанно отключить. Упоминаем pre-commit-хуки как бонус.

Практика: ПодключаемESLint+Prettier, настраиваем конфиг под проект, прогоняем автофикс, разбираем 2–3 предупреждения руками (например, неиспользуемые переменные, опасные сравнения). Проверяем, что в редакторе подсказки работают, и добавляем npm-скрипт lint.

8.3. Тема: Как устроить проект: модули и компоненты. Библиотеки.

Теория: От «кучи файлов» к понятной структуре. Договариваемся о слоях: базовые утилиты/константы, «фичи» (конкретные куски логики), UI-компоненты, страницы; избегаем циклических зависимостей. Принципы: одна зона ответственности, «тонкие» страницы, «толстые» фичи. Обсуждаем, как это упрощает навигацию по проекту и ревью.Разбираем стратегию «строим из своих модулей» vs «подключаем библиотеку». Зачем библиотеки: ускоряют разработку, закрывают нетривиальные куски (a11y-компоненты, даты, роутинг). Какие виды встречаются и чем отличаются

Практика: Рефакторим учебный проект: заводим папки src/components, src/features, src/lib, src/pages, раскладываем существующий код, называем модули понятно. Проверяем, что импорты через алиасы не сломались, запускаем сборку и быстрый регресс-чек экранов. Используем библиотеки для упрощения разработки.

Модуль 9: React и TypeScript.

9.1. Тема: React: первый компонент.

Теория: Почему React: декларативный подход и компоненты как «кирпичики» интерфейса. JSX — не шаблон, а синтаксис поверх JS: выражения в {}, один корневой элемент, className вместо class. Рендер корневого компонента, иерархия и композиция. Как мыслить компонентами: «какие данные меняются?» и «кто владеет состоянием?».

Практика: Создаём проект на Vite+React, пишем первый компонент (карточка приветствия) и подключаем его в App. Делаем маленькую композицию: заголовок, карточка, кнопка — как три компонента. Критерий: код читается, пропсы именованы понятно, нет «магических» строк.

9.2. Тема: React: состояние и события.

Теория: Состояние (useState) как источник правды для UI: меняем данные — React сам перерисует нужное. Обработчики событий в JSX (onClick, onChange) и однонаправленный поток данных. Принцип «мини-

мального состояния»: храним только то, что меняется, остальное — производное.

Практика: Реализуем счётчик, переключатель темы и простой фильтр списка: храним ввод пользователя и/или чекбоксы в состоянии, рендерим отфильтрованный набор. Проверяем, что при обновлении значения интерфейс меняется мгновенно и предсказуемо.

9.3. Тема: React: эффекты и загрузка данных.

Теория: Когда нужен useEffect: синхронизация с внешним миром (загрузка данных, подписки, таймеры). Зависимости эффекта и «утечки»: что будет, если список зависимостей неправильный. Очистка эффектов (return) и защита от гонок при быстрой смене параметров.

Практика: Компонент, который грузит список карточек по useEffect: показываем загрузку, пусто, ошибку; отменяем «устаревший» запрос при смене фильтра. Критерий: нет двойной загрузки, нет «мерцаний», кнопка «повторить» работает.

9.4. Тема: Введение в TypeScript.

Теория: Зачем статические типы в вебе: ловим ошибки до запуска (не тот ключ в объекте, неверная сигнатура функции). Базовые типы (string/number/boolean), массивы/объекты, литеральные и объединения ('asc' | 'desc'), typeinference (TS сам выводит тип) и ужесточение (noImplicitAny). Типизация функций (аргументы/возврат), узкое место: данные из формы всегда строки — учимся ужимать типы (narrowing) через проверки/guards. Коротко о tsconfig (строгий режим) и как Vite/React «понимают» TS без ручной сборки.

Практика: Переписываем 3–4 утилиты с JS на TS:

- форматер цены (formatPrice(value: number): string),
- парсер query-параметров (строка → объект),
- сравнение по полю для сортировки (generic на уровне «минимально понятно»).

- специально допускаем пару ошибок — смотрим, как TS их подсвечивает.

В конце — добавляем noImplicitAny: true и исправляем предупреждения.

9.5. Тема: TypeScript в React-компонентах.

Теория: Props через interface/type, необязательные поля (?) и значения по умолчанию. Состояние с дженериками: useState<number>, useState<string | null> и зачем union-тип «значение или отсутствует». Типизациясобытий (ChangeEvent<HTMLInputElement>, MouseEvent<HTMLButtonElement>), коллбэков ((value: string) => void). Коротко: React.FC и почему не злоупотребляем, предпочтительнее явные пропсы. Работа с API-ответом: описываем только используемые поля, не тащим весь объект.

Практика: Переписываем компоненты под TypeScript.

9.6. Тема: ReactRouter: навигация между страницами.

Теория: Маршруты как отображение структуры контента: BrowserRouter, Routes/Route, ссылки Link, активные состояния. Динамические параметры для страниц деталей (/items/:id), переключение экранов без перезагрузки, 404 на всё остальное. Как держать URL и состояние синхронизированными.

Практика: Настраиваем 2–3 маршрута: «Главная», «Список», «Детали». Делаем меню, активное состояние ссылки, страницу 404. Проверяем прямой заход по адресу и обновление страницы (SPA-fallback для продсборки проговорим ещё раз).

9.7. Тема: React: формы и контроль ввода.

Теория: Контролируемые поля: значение живёт в состоянии, UI — лишь «проекция». Валидация: где проверять (на вводе/на отправке), как показывать ошибки, как не «застрелить» пользователя сообщениями каждую секунду. События onChange, onSubmit, предотвращение дефолтного поведения.

Практика: Форма подписки/контактов: инпуты (имя, email), правила (обязательность, формат), сообщения об ошибках под полем, дизейбл кнопки при ошибке. После успешной отправки — «спасибо» и очистка. Проверка клавиатурой (Tab/Enter).

9.8. Тема: React: списки и ключи элементов.

Теория: Рендер коллекций через .map и роль ключа: он должен быть стабилен (ID), а не индекс. Что будет, если ключи неверные: потеря фокуса, «прыгающие» значения полей. Лёгкие оптимизации: мемоизация производных данных вне рендера, разделение больших списков на страницы.

Практика: Каталог элементов (данные из API или мока): корректные ключи, фильтр/сортировка, пагинация. Демонстрируем баг с ключами по индексу и исправляем на стабильные ID. Чек-лист: при удалении/добавлении состояние соседних элементов не сбивается.

9.9. Тема: React: простые кастомные хуки.

Теория: Когда пора делать хук: если один и тот же «танец» повторяется в двух компонентах (загрузка, дебаунс, локальное сохранение). Правила хуков (вызовы на верхнем уровне), зависимые значения и возврат интерфейса хука (данные + действия).

Практика: ПишемuseDebounce для поля поиска и useFetch для загрузки списка; подключаем оба в двух разных компонентах (например, поиск на главной и на странице списка). Критерий: компоненты «похудели», логика переиспользуется, багов меньше.

9.10. Тема: Проверочная работа по итогам освоения модуля 9.

Практика: Собираем мини-SPA на ReactRouter:

- / — приветствие и быстрый поиск,
- /items — список с пагинацией/поиском,
- /items/:id — детальная карточка.

Храним поисковую строку в query-параметре, при монтировании читаем её и запускаем загрузку. Добавляем кнопки «назад»/«домой», страницу

404 и SPA-fallback для прод-сборки. Приёмочные критерии: прямой заход по /items/123 работает, «обновить» не ломает навигацию, ошибки читаемы.

Модуль 10: Итоговый проект.

10.1. Тема: Итоговый проект: постановка задачи.

Практика: Каждый обучающийся/команда оформляет короткое ТЗ (1 стр.): цель, сценарии пользователя, страницы, источники данных, чек-лист доступности, критерии готовности. Создаём репозиторий, заводим issues (задачи), распределяем роли. На выходе — защищаем идею проекта в 2–3 минуты.

10.2. Тема: Итоговый проект: структура и контент.

Практика: Рисуем карту сайта (скетч), составляем таблицу сущностей и полей, подбираем/готовим тексты и изображения (источники/лицензии). Верстаем простые wireframes (бумага/Figma). Коммитим «контентную основу» в проект.

10.3. Тема: Проектный спринт I: каркас и сетки.

Практика: Реализуем каркас всех страниц проекта: хедер/футер/контейнеры, базовая сетка. Проверяем адаптив на 3 брейкпоинтах. Коммитим с понятными сообщениями, открываем PR на ревью.

10.4. Тема: Проектный спринт I: адаптив и изображения.

Практика: Подключаем изображения и иконки; правим медиазапросы, делаем флюидную типографику с clamp(). Прогоняем Lighthouse: фиксируем первые метрики (в особенности LCP/CLS), заводим задачи на улучшение.

10.5. Тема: Проектный спринт I: модули и переиспользование.

Практика: Выделяем 2–3 общих компонента (карточка, кнопка, секция), соединяем со страницами. Проверяем, что правка в одном месте меняет все вхождения. Обновляем документацию в README (как устроен проект).

10.6. Тема: Проектный спринт II: навигация и формы.

Практика: Настраиваем маршруты проекта, собираем меню/бургер. Реализуем 1 ключевую форму (по ТЗ): валидация, блокировка отправки, дружелюбные подсказки. Покрываем сценарии с клавиатуры.

10.7. Тема: Проектный спринт II: данные и API.

Практика: Подключаем 1–2 реальных эндпоинта или мок-сервер (на случай отсутствия ключей). Листинг/детали работают офлайн на моках, онлайн — на реальном API. Отлавливаем ошибки и retry.

10.8. Тема: Проектный спринт II: фильтры и поиск.

Практика: Добавляем строку поиска и фильтры; синхронизируем состояние с URL (восстанавливаем при загрузке). Проверяем сценарии: сброс/изменение фильтра — пагинация на первую страницу, отмена старого запроса.

10.9. Тема: Финальная подготовка к защите.

Практика: Финальный баг-фикс, сборка и деплой. Подготовка к презентации проекта.

10.10.Тема: Итоговая аттестация.

Практика: Презентация и защита итогового проекта и ответы на вопросы комиссии.

1.3 Планируемые результаты

Предметные:

- развит интерес к программированию у обучающихся;
- развиты умения применять идеи программирования в решении конкретных задач;
- знание и понимание основ веб-технологий;
- сформированы умения в области верстки и дизайна страниц;
- получены навыки программирования на JavaScript;
- сформированы умения работы с внешними данными(API);
- изучены основы разработки на React;
- получены проектные и презентационные навыки;
- простимулирована мотивация учащихся к получению знаний, оказана помощь в формировании творческой личности обучающегося.

Метапредметные:

- сформирована самостоятельность и творческий подход к решению задач с использованием средств вычислительной техники;
- воспитано упорство в достижении результата;
- воспитаны личностные качества: целеустремленность, настойчивость, самостоятельность, чувства коллективизма и взаимной поддержки, чувство такта.

Воспитательные:

- развиты коммуникативные компетенции: навыки сотрудничества в коллективе, малой группе (в паре), участия в беседе, обсуждении;
- сформированы и развиты информационные компетенции: навыки работы с различными источниками информации, умения самостоятельно искать, извлекать и отбирать необходимую для решения учебных задач информацию;
- сформировано представление о профессии сферы IT;
- сформирована самостоятельность и творческий подход к решению задач с использованием ПК и IT технологий;

- воспитано упорство в достижении результата;
- воспитаны личностные качества: целеустремленности, настойчивости, самостоятельности, чувства коллективизма и взаимной поддержки, чувство такта.

Воспитательная компонента курса реализуется через формирование активной жизненной позиции, бережного отношения к результатам своего труда и труду других, уважения к окружающим людям и природе. Важна поддержка инициативы, доверия к собственным силам и желания развиваться дальше. Учебная программа нацелена помогать ребенку раскрыть потенциал, расширить кругозор и заложить фундамент осознанного восприятия окружающей действительности.

Таким образом, планируемые результаты охватывают широкий спектр компетенций, значимых как для успешной учебы, так и для полноценной социализации ребенка в обществе.

2. Комплекс организационно-педагогических условий реализации общеразвивающей программы

2.1.Календарный учебный график на 2026-2027 учебный год

Таблица 2.

Год обучения	Сентябрь					октябрь					ноябрь					декабрь					январь					февраль					март					апрель					май				Всего учебных недель/часов	Всего часов по программе
дат	01.09 – 06.09	07.09 – 13.09	14.09 - 20.09	21.09 – 27.09	28.09 – 04.10	05.10 – 11.10	12.10 – 18.10	19.10 – 25.10	26.10 – 01.11	02.11 – 08.11	09.11 – 15.11	16.11 – 22.11	23.11 – 29.11	30.11 – 06.12	07.12 – 13.12	14.12 – 20.12	21.12 – 27.12	28.12 – 03.01	04.01 – 10.01	11.01 – 17.01	18.01 – 24.01	25.01 – 31.01	01.02 – 07.02	08.02 – 14.02	15.02 – 21.02	22.02 – 28.02	01.03 – 07.03	08.03 – 14.03	15.03 – 21.03	22.03 – 28.03	29.03 – 04.04	05.04 – 11.04	12.04 – 18.04	19.04 – 25.04	26.04 – 02.05	03.05 – 09.05	10.05 – 16.05	17.05 – 23.05		
неделя	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38		
1ВБ			4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2*		2 2	4	4	4	4	4	2*	4	4	4	4	4	4	4	4	4	4	4	4 2	35/136	136

Условные обозначения:

	Занятия по расписанию		Каникулярный период
	Приемная кампания		Промежуточная и итоговая аттестация
*праздничный день			

2.2. Условия реализации программы

Материально-техническое обеспечение

Требования к помещению:

- помещение для занятий, отвечающее требованиям СанПин для учреждений дополнительного образования;
- качественное освещение;
- столы, стулья по количеству обучающихся и 1 рабочее место для педагога.

Оборудование:

- Ноутбук педагога MSI M5-17LS– 1 шт.
- Ноутбуки ASUS FA707 - 13 шт.
- МФУ – Pantum BM5100DW – 1 шт.
- Интерактивная доска – 1 шт.
- Тележка для зарядки и хранения ноутбуков – 1 шт.
- Флипчарт магнитно-маркерный на треноге – 1 шт.
- Доска магнитно-маркерная настенная – 1 шт.
- Наушники 13 шт.

Информационное обеспечение

- Операционная система Windows
- Среда разработки WebStorm
- Среда разработки VSCode
- Браузер Yandex
- Текстовый редактор Notepad++

Кадровое обеспечение

Программа реализуется педагогом дополнительного образования – Красниковым Ильей.

2.3.Формы аттестации и оценочные материалы

Система отслеживания результатов обучающихся выстроена следующим образом:

- определение начального уровня знаний, умений и навыков;
- текущий контроль;
- промежуточная аттестация.
- итоговая аттестация.

Входящая диагностика по программе «WEB-разработка» проводится в начале освоения программы с целью выявления первоначального уровня знаний, умений и навыков обучающихся. Входная диагностика в форме опроса. Обучающемуся задается ряд общих вопросов, касающихся области алгоритмизации, информатики, программирования и информационных технологий в целом.

Текущий контроль осуществляется путем наблюдения, определения качества выполнения заданий, отслеживания динамики развития обучающегося. Способы проверки уровня освоения тем: опрос, выполнение упражнений, наблюдение. Формы проведения итогов по каждой теме и каждому разделу общеразвивающей программы соответствуют целям и задачам ДООП.

Таблица 3.

№	Раздел	Форма контроля	Критерии (1/2/3 балла)
1	**HTML-семантика и контент**	Практическая работа	1 — разметка фрагментарна, нарушена иерархия заголовков, alt формальны/отсутствуют. 2 — корректная базовая структура (header/main/footer), единичные недочёты в alt/заголовках. 3 — полноценная семантика (landmarks, оглавление), осмысленные alt, читабельность без стилей.
2	**Адаптивная вёрстка (Flex/Grid/изображения)**	Практическая работа	1 — ломается на мобильном, горизонтальный скролл. 2 — основные брейкпоинты и flex/grid применены, мелкие «прыжки». 3 — устойчивый макет, clamp() для шрифтов, srcset/picture, без CLS.
3	**Figma → код**	Практическая работа	1 — перенос стихийный, расхождения по типографике/отступам. 2 — ключевые секции перенесены близко к макету. 3 —

			системный перенос: токены/переменные, аккуратные отступы, совпадающая типографика.
4	**Git и публикация**	Практическая работа	1 — единичный коммит, нет README/деплоя. 2 — история из нескольких коммитов, деплой есть. 3 — атомарные коммиты, понятные сообщения, README с инструкциями, рабочий деплой.
5	**JavaScript: база (переменные/функции/массивы)**	Практическая работа	1 — решения частично работают, много ошибок типов/логики. 2 — задачи решены, встречаются мелкие огрехи. 3 — корректные функции, внятные имена, тесты в консоли на разные входы.
6	**DOM и события**	Практическая работа	1 — нестабильные обработчики, ломается фокус. 2 — базовые интеракции работают. 3 — делегирование событий, предсказуемый фокус, логика отделена от рендера.
7	**Формы и валидация**	Практическая работа	1 — нет связок label–input, ошибки непонятны. 2 — HTML5-валидация+несколько своих проверок. 3 — дружелюбные сообщения, aria-live, фокус на ошибочном поле, данные не теряются.
8	**Работа с API (fetch/состояния UI)**	Практическая работа	1 — запрос без обработки статусов/ошибок. 2 — loading/error/empty/successреализованычастично. 3 — надёжная обёртка, отмена «устаревших» запросов, ретрай/сообщения пользователю.
9	**Сборка и качество кода (Vite/ESLint/Prettier)**	Практическая работа	1 — сборка/линтер падают. 2 — собирается, отдельные предупреждения. 3 — чистый линт, алиасы путей, структура src согласована, README.
10	**React: основы (JSX/props/state)**	Практическая работа	1 — компоненты смешивают данные и представление, «магические» значения. 2 — базовая композиция работает. 3 — компонентное мышление, минимальное состояние, понятные пропсы.
11	**React Router и мини-SPA**	Практическая работа	1 — ломается при прямом заходе/обновлении. 2 — маршруты работают, мелкие баги. 3 — корректный SPA-fallback, активные ссылки, 404, состояние синхронизировано с URL.
12	**TypeScript (в проекте/React)**	Практическая работа	1 — типы расставлены точечно, множество any. 2 — типизированы основные сущности/пропсы. 3 — строгий режим, осмысленные интерфейсы, типизированные события/хуки.

Промежуточная аттестация обучающихся осуществляется в форме выполнения практического задания и оценивается по 10-бальной шкале, которая переводится в один из уровней освоения образовательной программы согласно таблице:

Таблица 4.

Баллы, набранные обучающимся	**Уровень освоения**
0–4 баллов	низкий
5–7 баллов	средний
8–10 баллов	высокий

Критерии и баллы промежуточной аттестации (10 баллов):

- Соответствие заданию - 2 (секции/функционал из условия).
- Семантика и доступность - 2 (заголовки, landmarks, alt, фокус виден).
- Адаптивность – 2 (без горизонтальной прокрутки; читаемая типографика, корректные брейкпоинты).
- Функциональность JS -3 (формы/события/DOM или работа с API; предсказуемые состояния UI).
- Сроки- 1 (сдано вовремя, в репозитории есть коммиты, деплой/превью доступно).

По завершении освоения дополнительной общеобразовательной общеразвивающей программы обучающиеся реализуется итоговая аттестация в виде защиты индивидуальных/групповых проектов. Темы для проектов обучающиеся выбирают на свое усмотрение при согласовании с педагогом или получают одну из тем проекта, подготовленных педагогом. Список предлагаемых тем проектов для выполнения обучающимися согласуется с учебной частью не позднее чем за 1 месяц до проведения итоговой аттестации по дополнительной общеобразовательной общеразвивающей программе «WEB-разработка».

Индивидуальный/групповой проект оценивается формируемой комиссией по 50-балльной шкале, которая переводится в один из уровней освоения образовательной программы согласно таблице 3:

Таблица 5

Баллы, набранные обучающимся	**Уровень освоения**
0–20 баллов	низкий
21–35 баллов	средний
36–50 баллов	высокий

Критерии и баллы итоговой аттестации (в сумме 50 баллов):

- Соответствие требованиям и сценарию пользователя- 6 (все заявленные страницы и сценарии работают).
- Самостоятельность - 5 (организация работы, осмысленные коммиты/PR, умение аргументировать решения).
- Дизайн и визуальная целостность - 6 (типографика, отступы, единый стиль, аккуратная работа с изображениями).
- Семантика и доступность - 6 (правильные заголовки/landmarks, фокус, читаемость, базовые aria/aria-live, клавиатурная навигация).
- Адаптивность и производительность - 6 (Lighthouse: без критических провалов; нет CLS; lazy-изображения, разумные размеры).
- Функциональность JS/React - 8 (компонентная структура, состояние, корректная работа форм, кастомные хуки при необходимости).
- Работа с данными - 5 (API/загрузка/ошибки/ретраи, синхронизация состояния с URL, устойчивость к «обновить страницу»).
- SEO/метаданные/OG-превью - 3 (title/description, og-теги, при необходимости — JSON-LD).
- Деплой и стабильность - 3 (рабочая прод-сборка, корректный SPA-fallback, понятный README).
- Сроки — 2 (готовый проект представлен вовремя).

Обязательность защиты. Защита проекта — обязательный этап завершения программы для получения Сертификата выпускника. При положительных результатах защиты обучающиеся получают сертификат, подтверждающий успешное освоение программы.

Состав комиссии (не менее 3 человек): в обязательном порядке входит педагог; приветствуется привлечение ИТ - профессионалов, представителей высших и других учебных заведений, администрации учебной организации.

Компонентами оценки индивидуального проекта являются (по мере убывания значимости): качество ИП, отзыв педагога, уровень презентации и защиты проекта. Если проект выполнен группой учащихся, то при оценивании учитывается не только уровень исполнения проекта в целом, но и личный вклад каждого из авторов. Решение принимается коллегиально.

2.4 Методические материалы

Перечень материалов, которые педагог использует при реализации программы. Список можно дополнять под конкретные темы и условия.

— Рабочая программа и календарно-тематическое планирование (КТП);

— Планы-конспекты/технологические карты занятий (формат 2×45 мин);

— Дидактические карточки заданий (уровни: минимум/норма);

— Рабочие листы для практических работ;

— Презентации (слайды) и видео по ключевым темам;

— Контроль (входной в виде опроса);

— Чек-лист выполнения практической работы и лист наблюдения педагога;

— Инструкции по установке ПО;

— Журнал посещаемости и ведомость оценивания.

Список литературы:

Основные источники

1. Дакетт, Дж. HTML и CSS. Разработка и дизайн веб-сайтов / пер. М. А. Райтмана. — Москва : Эксмо, 2020. — 480 с.
2. Дакетт, Дж. JavaScript и jQuery. Интерактивная веб-разработка / пер. М. А. Райтмана. — Москва : Эксмо, 2020. — 640 с. — ISBN 978-5-699-80285-2.
3. Хавербеке, М. Выразительный JavaScript. Современное веб-программирование / 3-е изд. — Санкт-Петербург : Питер, 2024. — 480 с. — ISBN 978-5-4461-1226-5.
4. Вероу, Л. Секреты CSS. Идеальные решения ежедневных задач / пер. Е. Шикаревой. — Санкт-Петербург : Питер, 2017. — 336 с. — ISBN 978-5-496-02082-4.
5. Чиннатамби, К. Изучаем React. 2-е изд. / К. Чиннатамби. — Москва : Бомбора (Эксмо), 2022. — 368 с. — ISBN 978-5-04-098028-4.
6. Мардан, А. React быстро. Веб-приложения на React, JSX, Redux и GraphQL / А. Мардан, М. Барклунд. — Санкт-Петербург : Питер, 2021. — 560 с. — ISBN 978-5-4461-0952-4.
7. Черный, Б. Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений / Б. Черный. — Санкт-Петербург : Питер, 2021. — 352 с.
8. Файн, Я. TypeScript быстро / Я. Файн, А. Моисеев. — Санкт-Петербург : Питер, 2021. — 540 с.

Дополнительные источники

1. Вейдт, В. П. О воспитании подростков: в помощь классному руководителю : методическое пособие / В. П. Вейдт. — Калининград : Изд-во Калининградского областного института развития образования, 2019. — 140 с.

2. Сэнд, У. Hello World! Занимательное программирование / У. Сэнд, К. Сэнд. — Санкт-Петербург : Питер, 2016. — 400 с.
3. Джексон, Н. Классный учитель: Как работать с трудными учениками, сложными родителями и получать удовольствие от профессии / Н. Джексон. — Москва : Альпина Диджитал, 2015. — 332 с.

Интернет-источники

1. Документация для веб-разработчиков [Электронный ресурс]. — URL: https://doka.guide/ (дата обращения: 04.08.2025).
2. Дополнительная документация для веб-разработчиков [Электронный ресурс]. — URL: https://webreference.com/ (дата обращения: 04.08.2025).
3. Информационный ресурс для веб-разработчиков [Электронный ресурс]. — URL: https://developer.mozilla.org/ru/ (дата обращения: 04.08.2025).
4. Современный учебник JavaScript [Электронный ресурс]. — URL: https://learn.javascript.ru/ (дата обращения: 04.08.2025).
5. Сайт с библиотеками для приложений JS [Электронный ресурс]. — URL: https://www.npmjs.com/ (дата обращения: 04.08.2025).
6. Официальный сайт NodeJS [Электронный ресурс]. — URL: https://nodejs.org/ (дата обращения: 04.08.2025).
7. Сайт TypeScript [Электронный ресурс]. — URL: https://www.typescriptlang.org/ (дата обращения: 04.08.2025).
8. Документация React [Электронный ресурс]. — URL: https://react.dev/ (дата обращения: 04.08.2025).
9. Система контроля версий [Электронный ресурс]. — URL: https://github.com/ (дата обращения: 04.08.2025).
10. Официальный сайт Git [Электронный ресурс]. — URL: https://git-scm.com/ (дата обращения: 04.08.2025).
11. Книга по работе с Git и Github [Электронный ресурс]. — URL: https://git-scm.com/book/ru/v2 (дата обращения: 04.08.2025).