Министерство образования и науки Республики Татарстан

Государственное автономное профессиональное образовательное учреждение «Набережночелнинский педагогический колледж»

КУРСОВАЯ РАБОТА

на тему

РАЗРАБОТКА САЙТА ОБУЧАЮЩЕГО РЕСУРСА ДЛЯ ИЗУЧЕНИЯ ОСНОВ СТИЛИЗАЦИИ ТЕКСТА С ПОМОЩЬЮ CSS

Специальность 09.02.07 Информационные системы и программирование

| Студент группы И-205 | \_\_\_\_\_\_\_\_\_/ Т.Р. Баев |
| --- | --- |
| Научный руководитель | \_\_\_\_\_\_\_\_\_/ Я. С. Голубева |

Набережные Челны, 2022

Оглавление

[Введение 3](#_heading=h.gjdgxs)

[1. Теоритическая часть 5](#_heading=h.30j0zll)

[1.1 Введение в предметную область 5](#_heading=h.1fob9te)

[1.2 Программные средства и технологии разработки 6](#_heading=h.3znysh7)

[1.2 Требования к интерактиву 9](#_heading=h.2et92p0)

[2. Практическая часть 11](#_heading=h.tyjcwt)

[2.1 Постановка задачи 11](#_heading=h.3dy6vkm)

[2.2 Разработка дизайна сайта 12](#_heading=h.1t3h5sf)

[2.3 Разработка сайта 15](#_heading=h.4d34og8)

[2.4 Тестирование сайта 23](#_heading=h.17dp8vu)

[Заключение 24](#_heading=h.3rdcrjn)

[Список используемой литературы 25](#_heading=h.26in1rg)

[Приложения 26](#_heading=h.lnxbz9)

# **ВВЕДЕНИЕ**

В настоящее время спрос на IT-профессии стабильно растет: с 2010 по 2020 год число вакансий в данной сфере на hh.ru увеличилось на 476%. Сегодня на hh.ru можно найти около 100 000 актуальных вакансий в ИТ. И самые востребованные среди них — программисты (38% от всех вакансий в отрасли). Но в программировании есть масса направлений, и одна из самых простых – это фронтенд-разработка.

Фронтенд-разработка — это создание пользовательского интерфейса на клиентской стороне веб-сайта или приложения. Это всё, что видит пользователь, когда открывает веб-страницу, и с чем он взаимодействует: кнопки, баннеры и анимация. В веб-разработке к технологиям фронтенда относятся HTML, CSS, JavaScript. Одна из этих технологий является каскадная таблица стилей CSS. CSS — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

CSS имеет большое количество свойств и правил, и самое тяжёлое – это то, что все эти правила нужно учить, чтобы уметь быстро и оперативно работать над проектом. Для того чтобы облегчить учёбу и сделать её более веселой, можно обучаться в игровом формате.

Цель курсового проекта состоит в создании простого обучающего ресурса, который позволит изучить свойства и правила стилизации текста в CSS. Сайт должен иметь современный адаптивный дизайн для удобного изучения.

Исходя из поставленной цели, были определены следующие задачи:

1. Провести общий анализ проектирования и создания сайтов с подобным функционалом;
2. Определить функциональные возможности Web-сайта;
3. Разработать сайт;
4. Наполнить контентом весь сайт.

Данный сайт был разработан с помощью таких средств разработки, как Visual Studio Code, Figma, так же использовалась библиотека Brython для взаимодействия языка Python с браузером. Так же была выполнена адаптация под разные разрешения, сайт функционирует как на стационарных компьютерах, так и на мобильных устройствах.

Сайт будет полезен начинающих(junior) фронтенд-разработчиков, которые только начинают изучать языки, требуемые для фронтенд-разработки, а также для более профессиональных программистов, которые ещё не работали с каскадной таблицей стилей. Начинающим фронтенд-разработчикам будет полезно пройти обучающую игру, для лёгкого старта изучения CSS.

# **1. ТЕОРИТИЧЕСКАЯ ЧАСТЬ**

## 1.1 Введение в предметную область

Разрабатываемый сайт должен представлять собой открытый образовательный ресурс в виде небольшой игры, состоящей из теоретической части и практической части.

Открытые образовательные ресурсы - это учебные, обучающие и исследовательские материалы на любом носителе – цифровом или ином, которая разрешает бесплатный доступ, использование, адаптацию или распространение другими лицами без ограничений.

Отличным примером образовательного ресурса будет сайт Flexbox froggy(<https://flexboxfroggy.com/>). На сайте присутствует интерактив в виде игры. Главная задача пользователя в этой игре передвинуть лягушек на кувшинки, используя CSS правило display со свойством flex. На ресурсе отлично показана работа с данным правилом, а обучение проходит в лёгкой и понятной форме (см. Рисунок 1).

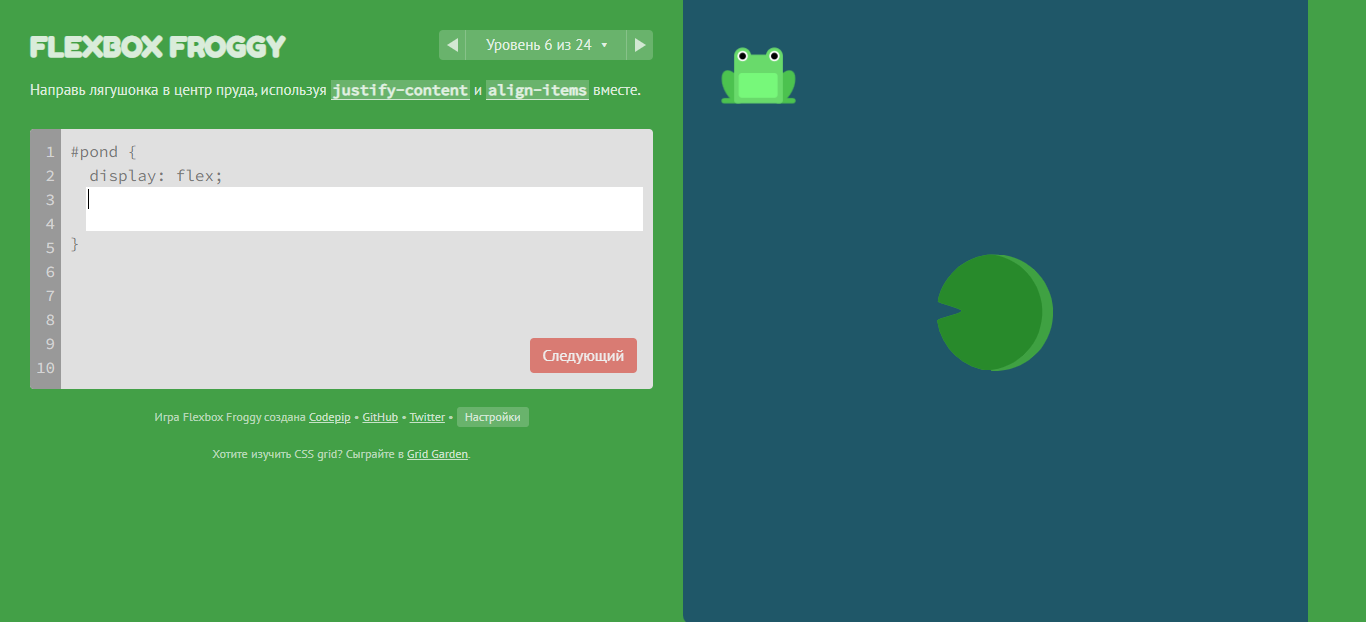


Рисунок 1. Обучающая игра Flexbox Froggy

Другим примером образовательного ресурса будет сайт HTML Academy(<https://htmlacademy.ru/>). На данном образовательном ресурсе можно купить онлайн курсы для изучения frontend и backend разработки. После покупки курсов, можно приступить к их прохождению. Прохождение курсов на HTML Academy проходят в формате, где пользователь сначала читает небольшую теорию (см. Рисунок 2), затем выполняет практическое задание, основанное на теории, которую прочитал пользователь. Страница практического задания состоит из редактора кода в левой части страницы, и двух окошек в правой части страницы. В верхнем окне расположен пример, который нужно повторить, в нижнем окне отображается результат кода редактора, который расположен в левой части страницы (см. Рисунок 3).

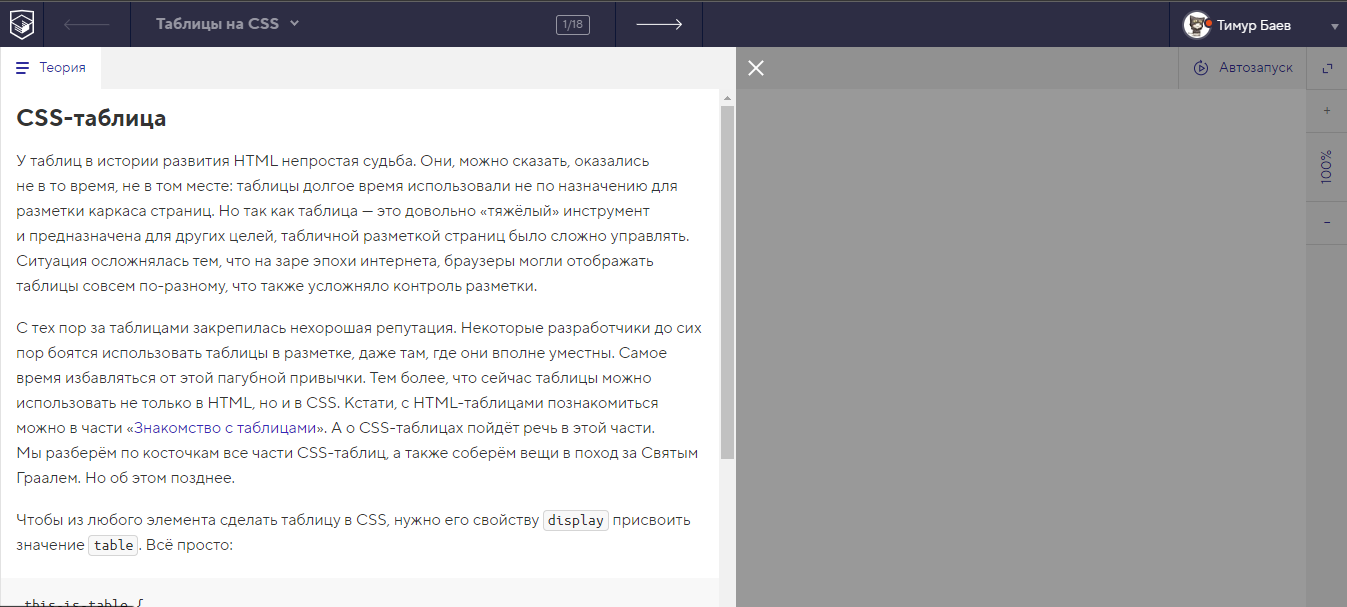


Рисунок 2. Блок с теорией в HTML Academy

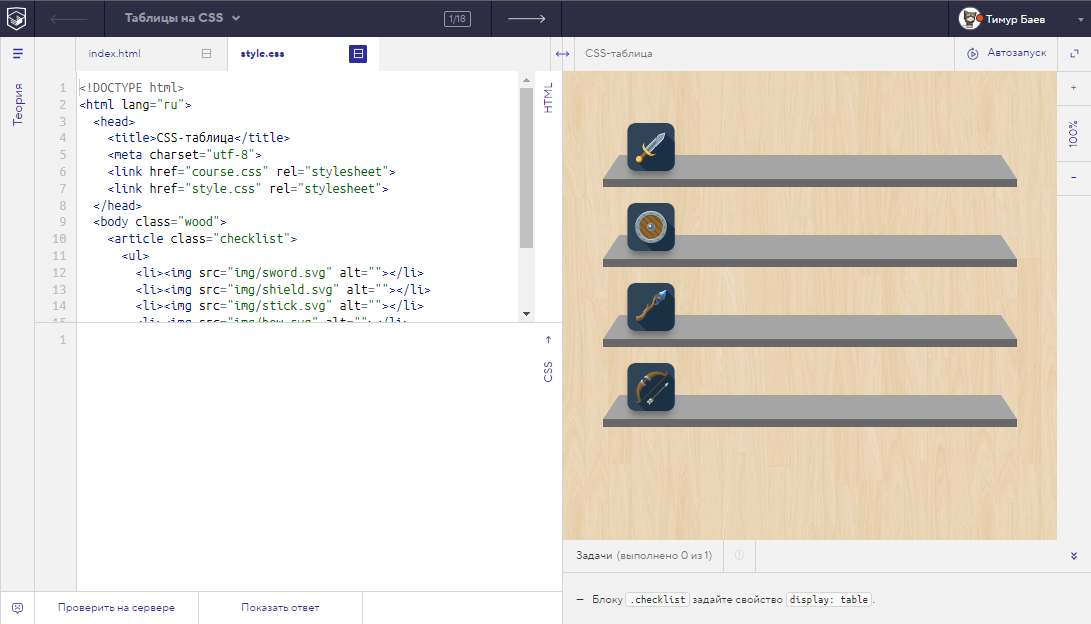


Рисунок 3. Блок с практикой в HTML Academy

Цель будущего сайта обучить пользователя использованию свойств и правил для стилизации текста в CSS.

Сайт должен состоять из одной страницы. В ней присутствует шапка, главная часть и подвал. В главной части располагается сама игра. Окно с игрой должно выглядеть как окно компьютера или другого устройства, в левой части компьютера должна располагаться теория с возможностью написать код CSS в текстовое поле, а на правой стороне должно быть два окна, в первом окне пример, который следует повторить, во втором результат работы кода, который пишет пользователь.

## 1.2 Программные средства и технологии разработки

Для разработки ресурса потребовались следующие программные средства и технологии разработки:

HTML (от англ. Hyper Text Markup Language - «язык гипертекстовой разметки») - стандартный язык разметки документов во Всемирной паутине. Большинство web-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа в удобной для человека форме.

Текстовые документы, содержащие разметку на языке HTML (такие документы традиционно имеют расширение.html или.htm), обрабатываются специальными приложениями, которые отображают документ в его форматированном виде. Такие приложения, называемые «браузерами» или «интернет - обозревателями», обычно предоставляют пользователю удобный интерфейс для запроса web - страниц, их просмотра (и вывода на иные внешние устройства) и, при необходимости, отправки введённых пользователем данных на сервер.

CSS (англ. Cascading Style Sheets - каскадные таблицы стилей) формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

CSS используется создателями web - страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих web-страниц. Основной целью разработки CSS являлось разделение описания логической структуры web-страницы от описания внешнего вида этой web-страницы. Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в web-разработке. Это прототипно – ориентированный сценарный язык программирования. Является диалектом языка ECMAScript.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности web - страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

Для создания вайфрейма и дизайна сайта я использовал онлайн-сервис для разработки интерфейсов и прототипирования с возможностью организации совместной работы в режиме реального времени Figma. Минимальные системные требования Figma:

Операционная система: macOS 10.12 (macOS Sierra) или более поздние версии; Windows 8 или более поздние версии с 64-битной средой. Браузер: Chrome 58+, Firefox 57+, Safari 11+, Microsoft Edge 16+. Видеокарта: Intel HD Graphics 4000, NVidia 330m, ATI Radeon HD 4850. Но Figma так же можно использовать онлайн.

Для создания верстки сайта, а так же, при работе с Python 3 я использовал редактор исходного кода Visual Studio Code. Минимальные системные требования Visual Studio Code:

Операционная система: Windows (7, 8 и 10), macOS X 10.11 или Linux (Ubuntu, Debian и Redhat). Процессор: 1,6 ГГц или выше. Память: 1 ГБ ОЗУ. Хранилище: 500 МБ свободного дискового пространства. Microsoft .NET Framework 4.5.2 также требуется для VSCode.

В Visual Studio Code я использовал плагин HTML CSS Support v1.13.1(см. Рисунок 3), который выдаёт контекстное меню с продолжением команды, для помощи в написании кода и более эффективной верстке. Что бы его установить, нужно использовать диалоговое окно «Управление расширениями». Чтобы открыть диалоговое окно «Управление расширениями», нужно выбрать Расширения > Управление расширениями. Или ввести строку «Расширения» в поле поиска, а затем выбрать «Управление расширениями». Затем в поиске набрать нужное расширение, затем выбрать его и нажать кнопку «установить» (см. Рисунок 4).

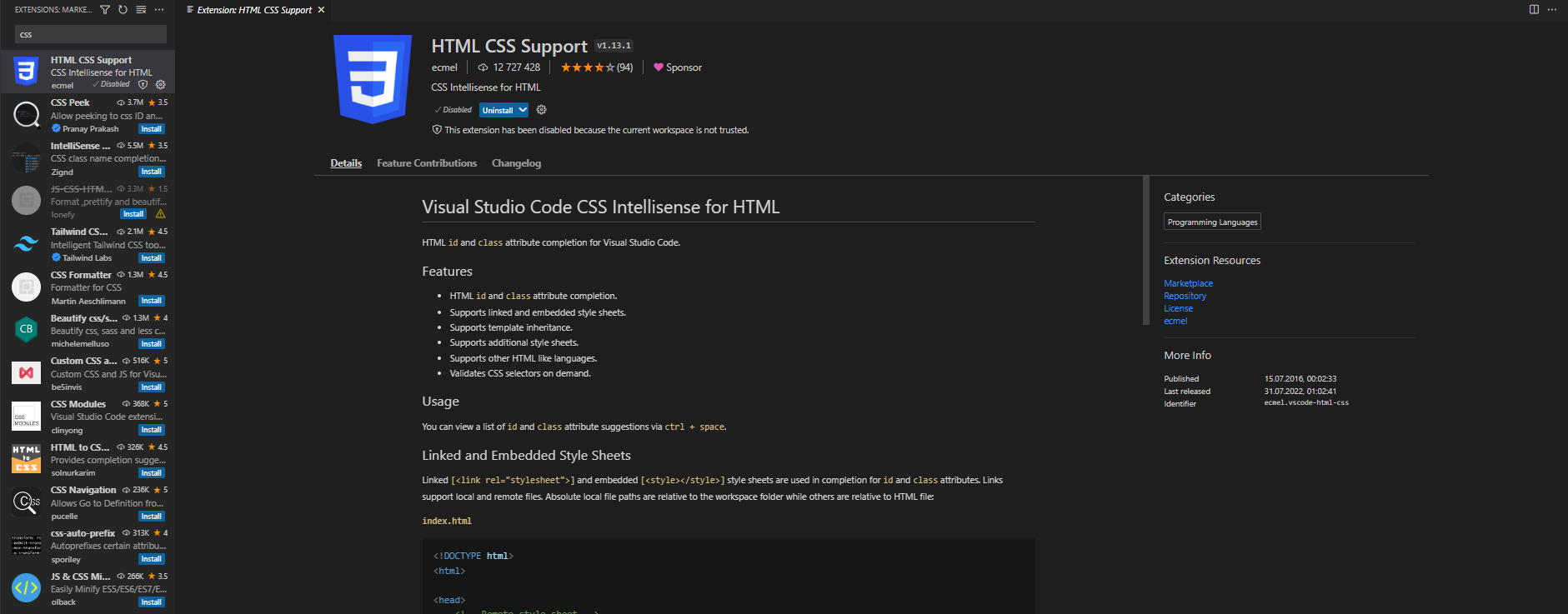


Рисунок 4. Расширение HTML CSS Support

Библиотека Brython для осуществления переключения по уровням, а также заменой уровней путём изменения текста. Brython призван заменить JavaScript в качестве языка сценариев для Интернета. Таким образом, это реализация Python 3, адаптированная к среде HTML5, то есть с интерфейсом к объектам и событиям DOM.

Используя Python 3, удалось реализовать систему смены уровней, не выходя из сайта, ни сменяя страницы. Все уровни хранятся в файле JSON. JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми, и прекрасно взаимодействует с разными языками программирования.

## 1.2 Требования к интерактиву

Необходимо создать структуру (шаблон) сайта, состоящую из следующих элементов:

1. «шапка» (header). В данном блоке необходимо расположить логотип;
2. «тело» (main) блок отображения информации, выбранный из меню;
3. «подвал» (footer) сайта. В данном блоке необходимо поместить контактную информацию о предприятии и логотип.

Помимо требований к данным, необходимо определить требования к эргономике программы. Эргономичность (юзабилити) — способность продукта быть понимаемым, изучаемым, используемым и привлекательным для пользователя в заданных условиях; свойство системы, продукта или услуги, при наличии которого конкретный пользователь может эксплуатировать систему в определенных условиях для достижения установленных целей с необходимой результативностью, эффективностью и удовлетворённостью.

Интерфейс веб-сайта должен быть максимально простым, что позволит пользователю быстрее адаптироваться, увеличивает скорость его работы, уменьшит вероятность совершения ошибок; Дружественным, чтобы пользователь всегда мог знать на какой стадии процесса он находится; Естественным, то есть не вынуждать пользователя существенно изменять привычные для него способы решения задачи. Интерфейс должен отражать функциональность программы и давать возможность успешной работы пользователям различной квалификации, в том числе и людям с ограниченными возможностями.

Вдобавок у пользователя должна быть возможность писать CSS свойство в текстовое поле и увидеть его действие напрямую на сайте, не выполняя никаких лишних манипуляций и без длительного ожидания. Притом, при нажатии переключателя в виде стрелок, должна происходить смена уровня. При смене уровня, у некоторых элементов сайта должна осуществляться смена текстового значения, и изменение стилевого оформление.

Пока проект будет храниться только на нашем компьютере, и о существовании сайта никто не будет знать. Чтобы сайт появился в Интернете и его увидели другие пользователи, нужно разместить эти файлы в сети. Для этого нужно:

1. Зарегистрировать домен.
2. Заказать хостинг.
3. Привязать домен к хостингу.
4. Разместить сайт на хостинге.

Хостинг — это услуга по предоставлению ресурсов на сервере, иначе говоря, аренда дискового пространства, где будут храниться все необходимые для правильного функционирования сайта файлы и данные. Серверы постоянно находятся в сети и подключены к бесперебойному электропитанию. Благодаря этому размещенные на них сайты доступны круглосуточно. Таким образом, хостинг — это аренда места и мощностей.

# **2. ПРАКТИЧЕСКАЯ ЧАСТЬ**

## 2.1 Постановка задачи

Для начала, проведём общий анализ проектирования и создание сайтов с подобным функционалом. Из нужных нам сайтов выделяем несколько с простым и хорошим функционалом. А также выделяем примерные системные требования, которые понадобятся со стороны клиента и сервера.

Системные требования - это описание примерных характеристик, которым должен соответствовать компьютер для того, чтобы на нём могло использоваться какое-либо определённое программное обеспечение. Эти характеристики могут описывать требования как к аппаратному обеспечению (тип и частота процессора, объём оперативной памяти, объём жёсткого диска), так и к программному окружению (операционная система, наличие установленных системных компонентов и сервисов и т.п.). Обычно такие требования составляются производителем или автором программного обеспечения.

Системные требования для работы программного продукта должны быть следующими:

Со стороны сервера:

* сервер с возможностью использования Python скриптов

Со стороны клиента:

* браузер Mozilla Firefox 5.0+, Chrome 13+, Opera 11+, Internet Explorer 8.0+
* наличие сети интернет;
* процессор с тактовой частотой, 1.6 ГГц, не менее;
* оперативная память 128 Мбайт и выше;
* жесткий диск (200 Мбайт свободного места) и выше;
* клавиатура;
* манипулятор мышь;

Для начала, подберём все возможные стили текста, которые существуют в CSS и составил небольшую таблицу (см. Рисунок 1). Среди всех существующих, выберем 11 самых нужных, которые потребуются при разработке сайта.

Таблица 1. Некоторые свойства текста в CSS

| **Свойство** | **Значение** | **Описание** | **Пример** |
| --- | --- | --- | --- |
| font-family | имя шрифта | Задает список шрифтов | P {font-family: Arial, serif} |
| font-style | normal italic oblique | Нормальный шрифт Курсив Наклонный шрифт | P {font-style: italic} |
| font-variant | normal small-caps | Капитель (особые прописные буквы) | P {font-variant: small-caps} |
| font-weight | normal lighter bold bolder 100–900 | Нормальная жирность Светлое начертание Полужирный Жирный 100 — светлый шрифт, 900 — самый жирный | P {font-weight: bold} |
| font-size | normal pt px % | нормальный размер пункты пикселы проценты | font-size: normal font-size: 12pt font-size: 12px font-size: 120% |

## 2.2 Разработка дизайна сайта

После начнем создавать вайрфрейм проекта. В вайрфрейме присутствуют такие элементы, как шапка страницы, подвал страницы и главное окно по работе со стилями. В главном окне находится текстовое поле для записи стиля и текст для теории, который пока что задан текстом-заполнителем. Текст, у которого нужно будет повторить стили, и текст, на который будут применяться стили, записанные в текстовое поле.

Вайрфрейм — то простой способ визуализации дизайна приложения или веб-сайта, а также определения элементов нового проекта. Созданная простая схема позволит легко представить, как различные элементы будут взаимодействовать друг с другом и выявить потенциальные UX-проблемы.

Текст-заполнитель — это текст, который имеет некоторые характеристики реального письменного текста, но является случайным набором слов или сгенерирован иным образом. Его можно использовать для отображения образца шрифтов, создания текста для тестирования или обхода спам-фильтра.

Также, для показа стилей на текстах, в дальнейшем, мы будем использовать текст-заполнитель. Таким образом, пользователь не будет читать и отвлекаться на лишнюю информацию, будет легче показывать все стили текста.

На рисунке 5 представлен вайрфрейм сайта.

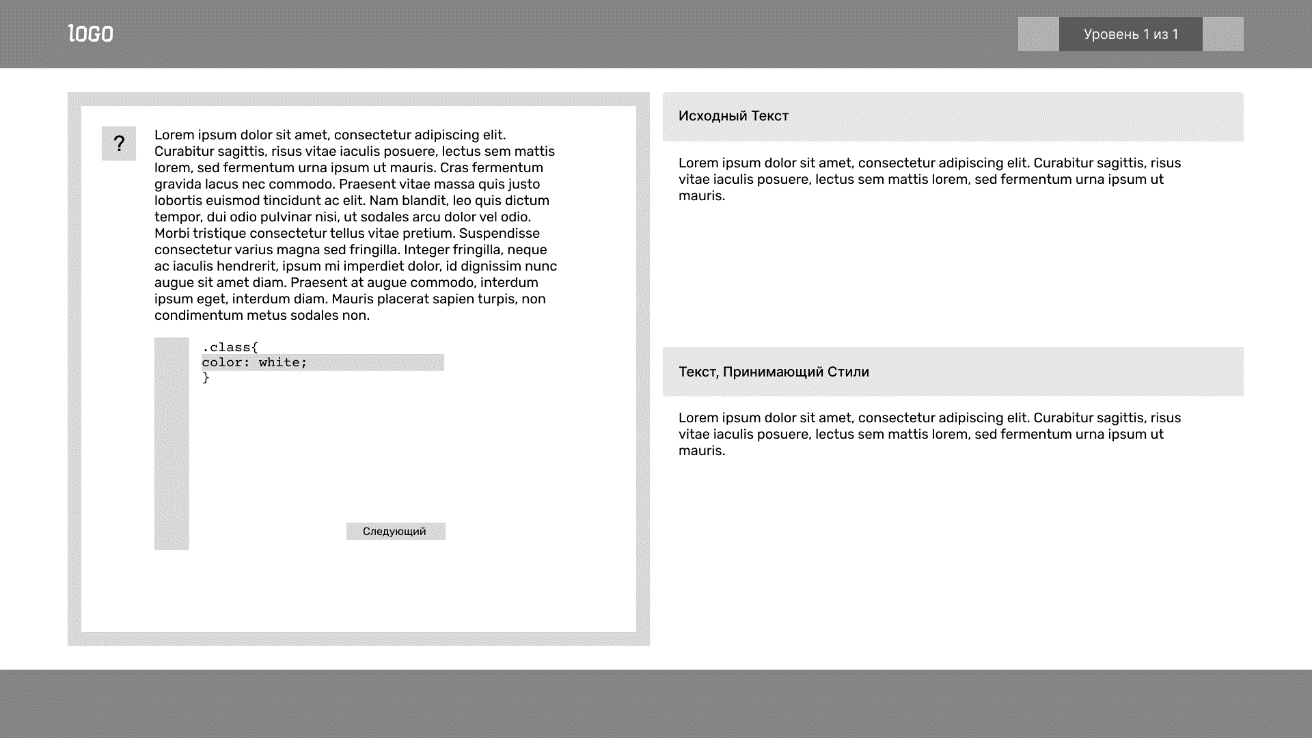


Рисунок 5. Вайрфрейм будущего веб-сайта

В процессе создания вайрфрейма, у проекта появилась идея и дальнейшее оформление проекта. Дизайн сайта будет выполнен в космическом стиле, и идея такая, что космонавт получает входящие сообщения, которые следует повторить, используя CSS.

Приступим к созданию дизайн-макета сайта. Для создания стилевого оформления, используем специальный веб-сервис под названием coloors.co. Сервис позволяет сгенерировать случайное стилевое оформление сайта.

Дизайн-макет сайта – это визуальный образ будущего сайта, разработанный с учетом технических возможностей HTML верстки. Такой макет является демонстрацией того, как визуально будет выглядеть ваш сайт после верстки и наполнения.

На рисунке 6 показан дизайн-макет будущего веб-сайта.

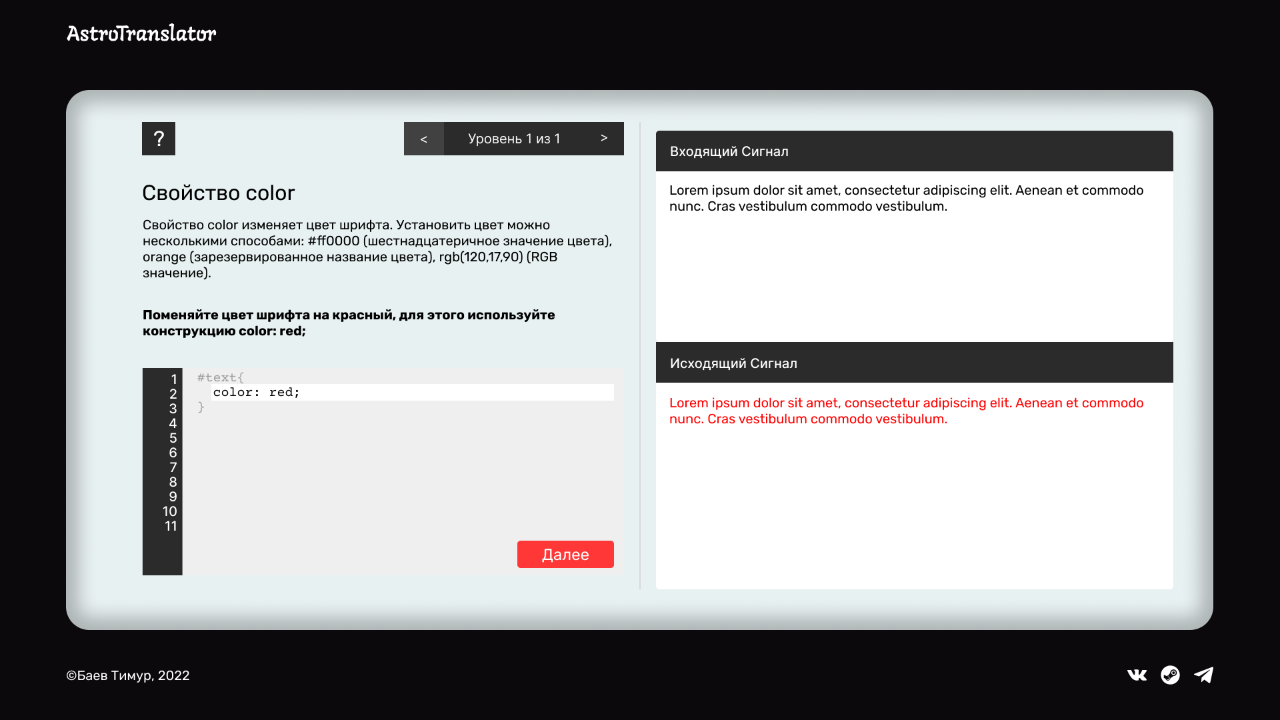


Рисунок 6. Дизайн-макет будущего веб-сайта

Функциональные возможности Web-сайта будут зависеть от библиотеки предназначеной для работы с сайтом Brython, так как библиотека является относительно новой, и не имеет полный контроль над сайтом.

Помимо этого, благодаря Python 3 будет возможно создать сайт с изменяющимися данными без использования серверных языков программирования и больших баз данных. Все уровни курсов будут храниться в файле типа JSON. С данными типом легко работать как человеку, так и машине.

## 2.3 Разработка сайта

Для начала, создадим файл index.html, в котором будет строиться вся верстка сайта. Построим семантическую верстку сайта, используя среду разработки Visual Studio Code.

Семантическая вёрстка — подход к разметке, который опирается не на содержание сайта, а на смысловое предназначение каждого блока и логическую структуру документа.

Создадим семантическую верстку блока с текстовым полем (см. Листинг 1). В данном блоке все построено так, чтобы у пользователя была возможность наблюдать, к чему он использует стили. Ещё можно заметить, что в данном блоке присутствует кнопка, для перехода на следующий уровень.

Листинг 1. Блок с текстовым полем.

<div class="coding">

<h2 hidden>Код</h2>

<div class="line-numbers">1 2 3 4 5 6 7 8 9 10</div>

<pre>#text {</pre>

<textarea id="code" autocapitalize="none" autofocus style="height: 25px; resize: none;"></textarea>

<pre>}</pre>

<button class="next-btn" disabled>Далее</button>

</div>

После создания верстки сайта, принимаемся за его стилизацию с помощью каскадных таблиц стилей CSS. Используя CSS можно сделать так, что на сайте будет один стиль на все элементы. Но это не удобно, так как если изменять внешний вид одного элемента, \придется переписывать стиль всего сайта. Для этого используем селекторы.

Селекторы – это выражения, которые говорят браузеру, к какому элементу HTML нужно применить те или иные свойства CSS, определённые внутри блока объявления стиля.

Таким образом, получилось стилизовать сайт. Следует отметить, что стили для блока с текстовым полем (см. Листинг 2) были отлично подогнаны под ширину экрана и сайт стал выглядеть, как показано на рисунке 7.

Листинг 2. Стили блока с текстовым полем.

.coding{

height: 260px;

position: relative;

padding: 10px 10px 10px 40px;

background-color: #e0e0e0;

border-radius: 4px;

overflow: hidden;

font-family: 'Source Code Pro', monospace;

font-size: 16px;

line-height: 1.5;

margin-top: 20px;

}

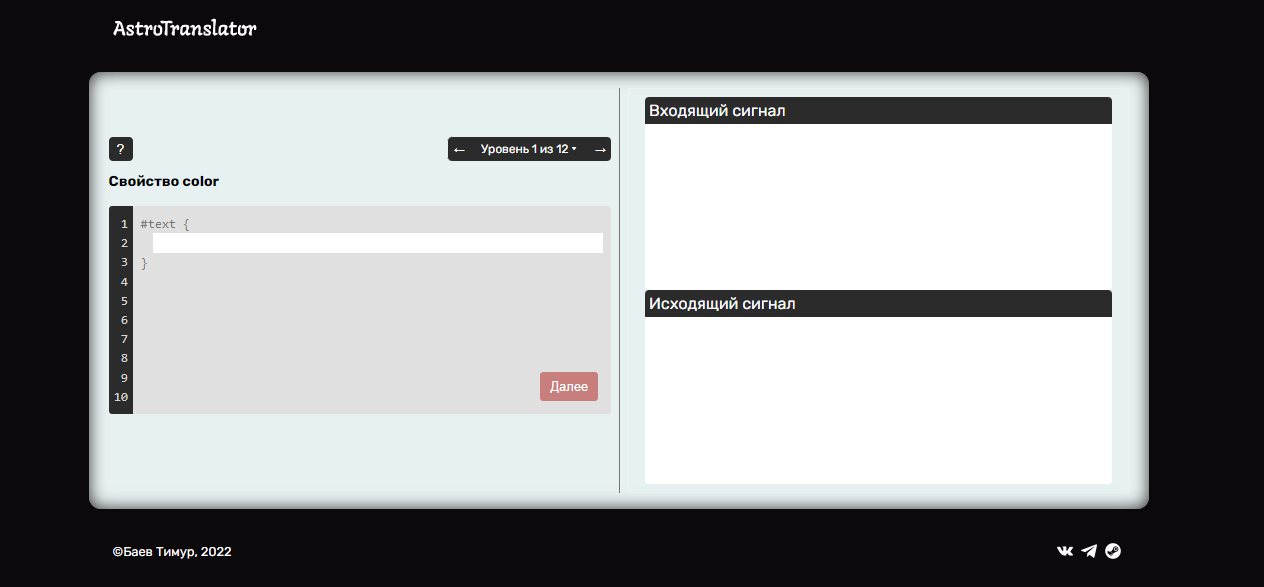
\

Рисунок 7. Верстка сайта

Однако, к сожалению, при уменьшении масштаба экрана текстовое поле выходит за рамки самого блока. Поэтому создадим адаптивность сайта.

Адаптивность сайта – это способность веб-ресурса корректно отображаться на устройствах с разными аппаратными и программными характеристиками.

В конечном итоге, удалось выполнить верстку сайта. Вдобавок, при изменении разрешения под планшетные устройства, происходит перестройка сайта, и теперь, когда ширина экрана становится 780px, то левый и правый блоки выстраиваются в одну колону (см. Рисунок 8).

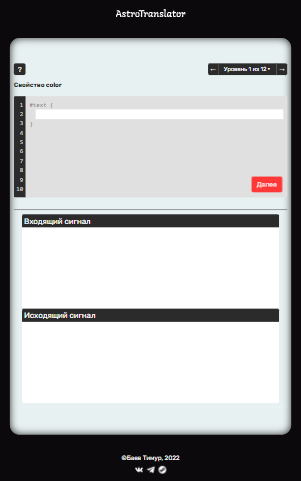


Рисунок 8. Адаптация под мобильные устройства

Сейчас имеется пустой сайт, но надо, чтобы на сайте были «Уровни». Сделать это можно множественными способами, например, использовать серверные языки программирования. Однако, можно обойтись и без них, используя объекты в JavaScript. Но есть одна проблема: написание уровней и скриптов на одном файле будет громоздким, и работать таким образом с данными очень сложно.

Поэтому было принято решение использовать специальный тип фалов JSON, который хранит в себе объекты или массив объектов. Вдобавок, с таким файлом легко работать, потому что там всё видно наглядно.

Во всяком случае, будет использоваться этот тип фалов для хранения данных. Но при работе с данным типом данных в JavaScript, можно столкнуться с проблемой: в JavaScript невозможно импортировать файл JSON.

Решением этой проблемы оказалось не очень простым, однако было принято решение использовать язык программирования Python. Язык программирования Python имеет огромное количество модулей, позволяющих работать с разными данными, и в разных областях разработки, а еще имеет удобную систему работы с файлами.

Но это не стало финальным решением проблемы, потому что единственный язык программирования, который воспринимают браузеры, является JavaScript. Опять же, нашлось решение и этой проблемы, для того чтобы браузер мог обработать Python скрипты, будет использоваться библиотека Brython.

Brython — это реализация Python3, написанная на JavaScript, которая позволяет писать код на Python для веба. По сути, это JavaScript-библиотека, которая преобразует ваш код на Python в эквивалентный JS и исполняет его в рантайме.

Началось написание Python скрипта, который переключает задания при нажатии стрелок или кнопки «Далее». Для начала, заполним тестовые данные в JSON файл. Будет использоваться массив объектов для хранения данных об определённых уровнях. В Python скрипте будем использовать систему индексации (см. Приложение 1 Листинг 3).

Индексация в языках программирования — это механизм для доступа к компоненте массива данных посредством ссылки на массив и посредством одного или более выражений, значения которых определяют позицию компоненты массива.

Индекс — элемент перечислимого множества, который указывает на конкретный элемент массива. Обычно является неотрицательным целым числом. В некоторых языках отрицательные индексы используются для отсчёта элементов в обратную сторону (начиная с конца массива).

Таким образом, удалось создать навигацию по уровням, а так же смену контента при смене уровня и сайт приобрел вид как на рисунке 9.

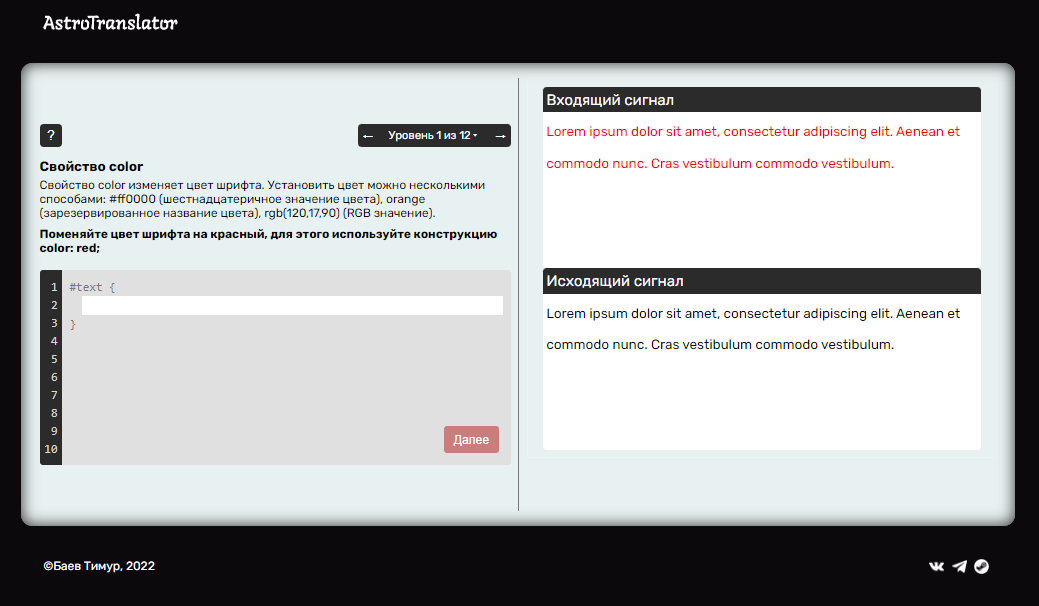


Рисунок 9. Финальный вид сайта

После окончания всей работы с Python, было создано пару скриптов на языке программирования JavaScript. Первый скрипт активирует кнопку «Далее» если пользователь правильно повторит стили с исходного сообщения (см. Листинг 4). Второй скрипт открывает меню с подсказкой, при нажатии на кнопку со знаком вопроса как показано на рисунке 10, а код представлен в листинге 5.

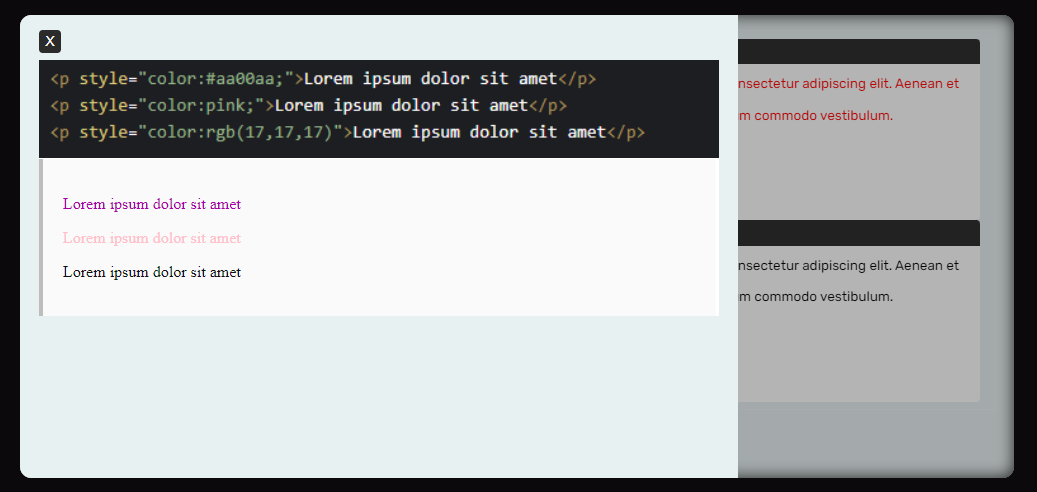


Рисунок 10. Меню с подсказкой

Листинг 5. Скрипт, показывающий подсказку

let helpBtn = document.querySelector('.help-button')

let hideBtn = document.querySelector('.hide-theory-button')

let theoryWrapper = document.querySelector('.theory-wrapper')

let theory = document.querySelector('.theory')

let levelIndicator = document.querySelector('.level-indicator')

let levelsWrapper = document.querySelector('.levels-wrapper')

helpBtn.onclick = function(){

theoryWrapper.classList.add('show-theory');

}

hideBtn.onclick = function(){

theoryWrapper.classList.remove('show-theory');

}

levelIndicator.onclick = function(){

levelsWrapper.classList.add('levels-wrapper-show')

}

document.addEventListener('mousedown', function(e){

if(e.target.closest('.levels-wrapper-show') === null){

levelsWrapper.classList.remove('levels-wrapper-show')

}

});

Так как Python скрипт прогружался не сразу, а с задержкой, было принято решение, создать небольшое ожидание в виде загрузки. Для начала создаём вёрстку экрана загрузки в HTML (См. Листинг 6). А также создём для него анимацию с помощью CSS (См. Листинг 7).

Листинг 6. Разметка экрана загрузки

<div class="loadingScreen">

<div>

<div class="item">

<i class="loader --1"></i>

<p>Загрузка</p>

</div>

</div>

</div>

Листинг 7. Анимация ухода экрана загрузки

@keyframes loading {

0%{

transform: translate(0, 0);

}

100%{

transform: translate(0, -100%);

}

}

И в итоге получился экран загрузки, который показан на рисунке 11.

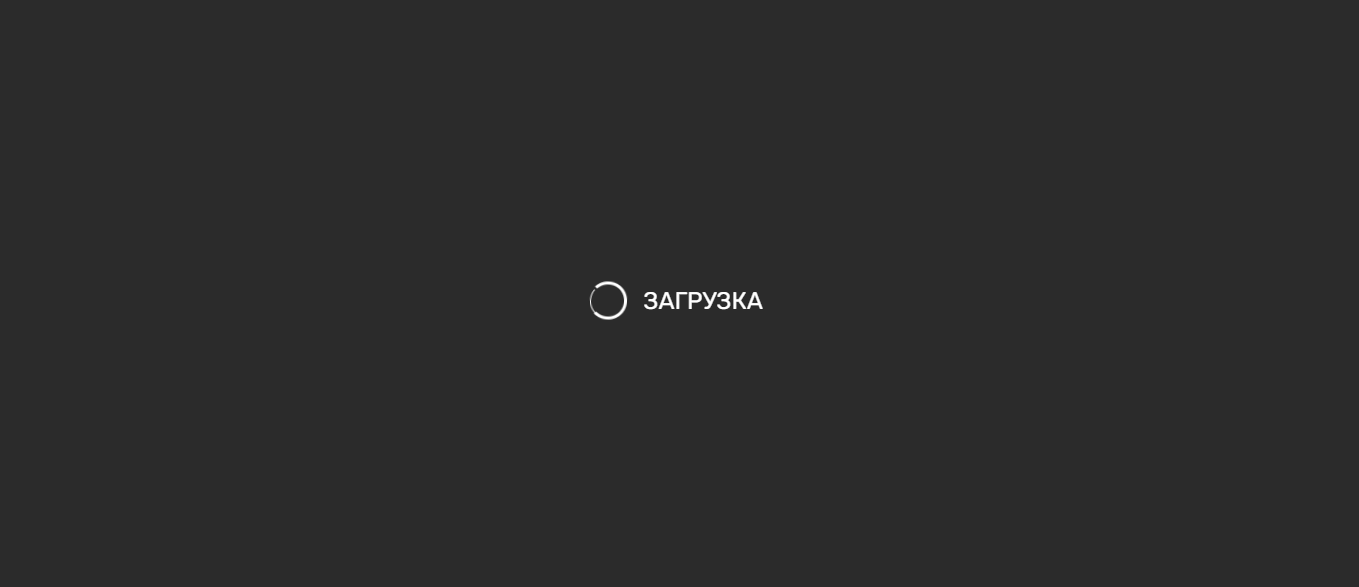


Рисунок 11. Экран загрузки сайта

После завершения всего сайта, приступаем к выкладыванию сайта на хостинг, благодаря чему сайт может находиться в интернете, а не на компьютере.

Для начала, нужно зарегистрировать домен, но это не самая дешевая операция. Так как хостинг предусматривает трату денег для покупки домена, воспользуемся бесплатным интернет хостингом beget.com, он показан на рисунке 12 (https://beget.com/ru), у которого есть акция: при регистрации, мне даётся хостинг на 1 месяц, но этот хостинг будет содержать шаблонный домен.

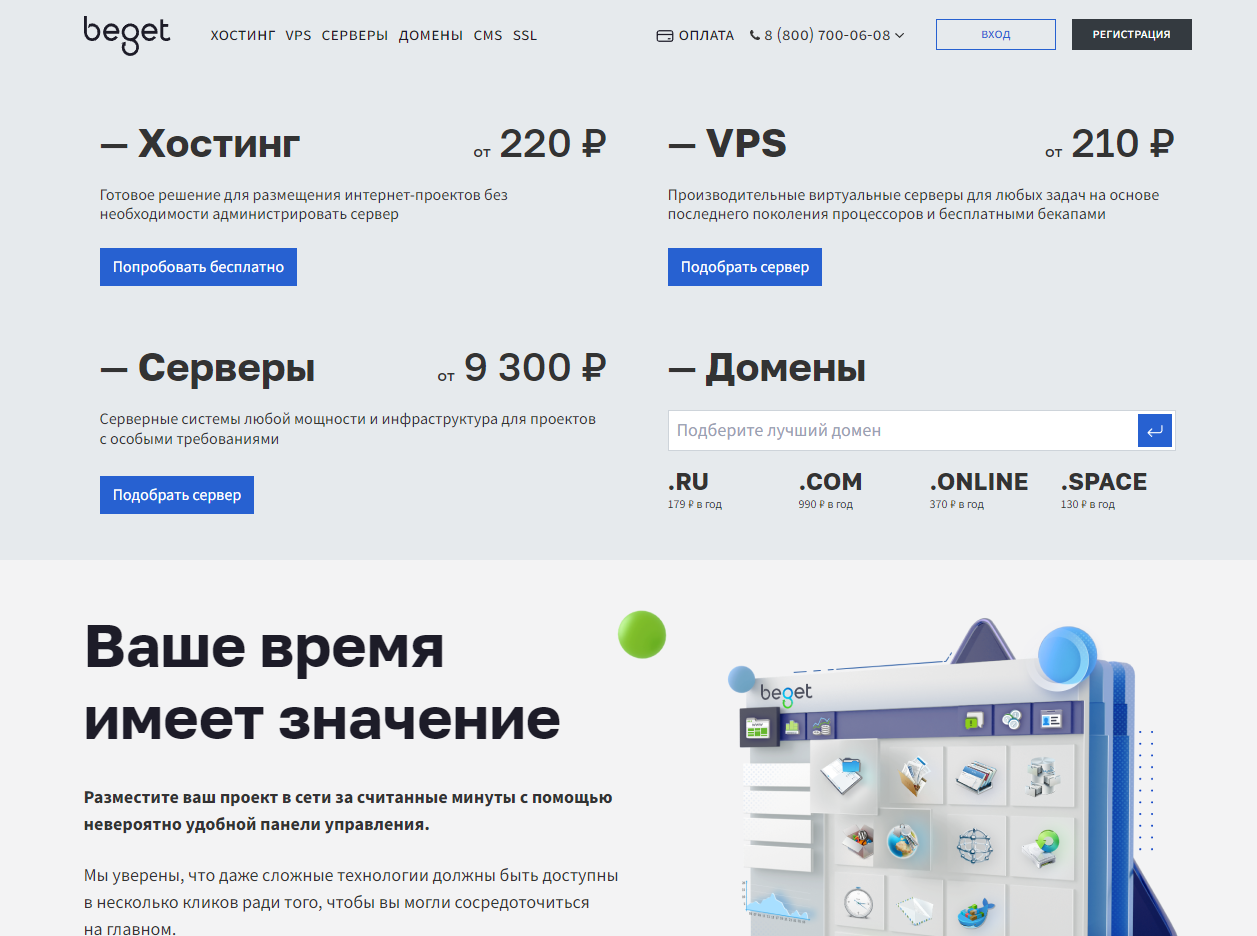


Рисунок 12. Хостинг сайтов beget

После получения хостинга, поместил свои файлы сайта в его директорию. Далее, привязываем свой шаблонный домен к своему хостингу. И в конце концов таким образом размещаем сайт на хостинге.

И в конечном итоге, сайт начал отображаться в браузере с ссылкой в адресной строке (см. Рисунок 13).

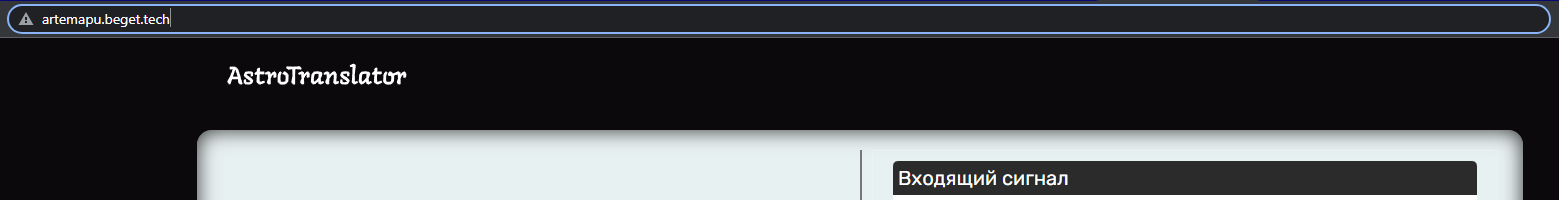


Рисунок 13. Отображение ссылки в адресной строке

## 2.4 Тестирование сайта

После полного создания сайта, было проведено его диагностическое тестирование, благодаря которому, удалось выявить явные недочёты сайта, и сложность прохождения курсов.

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Тестировщик — специалист, принимающий участие в тестировании компонента или системы. В его обязанность входит поиск вероятных ошибок и сбоев в функционировании объекта

Что бы протестировать сайт нужен новый взгляд, поэтому было принято решение найти тестировщика. Тестировщик нужен для того, чтобы процесс исправления ошибок происходил успешнее. В качестве тестировщика был взят студент 2 курса, который как раз изучает HTML и CSS свойства.

Тестировщик прошёл курсы, и не выявил некоторые недостатки, поэтому, после исправления недочётов, проект можно считать завершённым.

Некоторые процессы тестирования показаны в приложении 2.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения работы были изучены и закреплены знания по созданию и проектированию веб-сайтов, а также приобретены практические навыки по созданию веб-сайтов c использованием языка программирования JavaScript, Python с использованием библиотеки Brython.

Данный веб-сайт позволяет:

* Обучиться свойствам текста каскадных таблиц стилей CSS;
* Применить освоенные знания в практическом опыте.

В результате выполнения данного курсового проекта будет разработано полноценный веб-сайт «Обучающие курсы по стилизации текста в CSS», выполняющие все требуемые функции и поставленные задачи. Цель проекта была достигнута.

Были также решены следующие задачи:

* Изучен язык программирования Python
* Изучена библиотека Brython.

К сожалению, не получилось реализовать множество функций, потому что библиотека Brython относительно новая, и не поддерживает некоторые функции JavaScript. Но это только даёт возможность продолжить работу над проектом и усовершенствовать его со временем.

# **СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Информационный справочник — URL: http://htmlbook.ru/content/svoystva-teksta
2. Mozilla Developer Network или Сеть Разработчиков Мозилла (MDN) развивающаяся платформа для обучения Веб-технологиям и программному обеспечению, на которой основан Web, включая: Веб стандарты, на пример CSS, HTML, и JavaScript: https://developer.mozilla.org/ru/docs/Learn/CSS/Styling\_text/Fundamentals
3. «ИТ Шеф» Сайт о веб-разработке и программировании: https://itchief.ru/html-and-css/text
4. Официальная документация – справочник по библиотеке Brython — URL: https://brython.info/static\_doc/en/intro.html
5. HTML Academy: интерактивные онлайн-курсы по HTML, CSS и JavaScript: https://htmlacademy.ru/
6. Генератор цветовых решений: https://coolors.co/

# **ПРИЛОЖЕНИЯ**

Приложение 1

Листинг 3. Скрипт для смены уровня в Python

#Импорт нужных модулей

import json # Модуль json для интегрирования json-файла в программу

from browser import \* # Модуль browser для взаимодействия с html

#Открытие файла json со всеми уровнями

file = open('levels.json', 'r')

config = json.load(file)

#Выбор и присваивание в переменные всех нужных элементов сайта для работы с ними

lvlCounter = 0

totalLvl = len(config)

totalText = document.select('.total')[0]

nextBtn = document.select('.next-btn')[0]

levels = document.select('.level')

currentLvl = document.select('.current')[0]

totalNumsOfLvls = document.select('.total')[0]

leftArrow = document.select('.left-arrow')[0]

rightArrow = document.select('.right-arrow')[0]

hintCode = document.select('.hint')[0]

hintAlert = document.select('.hint')[1]

theorytext = document.select('.exercise p')[1]

exercise = document.select('.exercise b')[0]

caption = document.select('.caption')[0]

example = document["example"]

changingText = document["text"]

code = document["code"]

#начальные данные

totalText.text = totalLvl

hintCode.src = config[0]['hintCode']

hintAlert.src = config[0]['hintAlert']

theorytext.text = config[0]['theorytext']

exercise.text = config[0]['exercise']

caption.text = config[0]['caption']

example.text = config[0]['example']

example.style.cssText = config[0]['exampleStyle']

changingText.text = config[0]['changingText']

changingText.style.cssText = config[0]['changingTextStyle']

#Основные действия программы

#Метод для смены уровня назад

def lvlBack(event):

global currentLvl

global lvlCounter

if(lvlCounter == 0):

pass

else:

lvlCounter -= 1

currentLvl.text = lvlCounter+1

lvlGen()

#Метод для смены уровня вперед

def lvlAhead(event):

global currentLvl

global lvlCounter

global totalLvl

if(lvlCounter == totalLvl-1):

pass

else:

lvlCounter += 1

currentLvl.text = lvlCounter+1

lvlGen()

#метод для генерации уровня

def lvlGen():

global lvlCounter

global hintAlert

global hintCode

global theorytext

global caption

global example

global changingText

global config

hintCode.src = config[lvlCounter]['hintCode']

hintAlert.src = config[lvlCounter]['hintAlert']

theorytext.text = config[lvlCounter]['theorytext']

exercise.text = config[lvlCounter]['exercise']

caption.text = config[lvlCounter]['caption']

example.text = config[lvlCounter]['example']

example.style.cssText = config[lvlCounter]['exampleStyle']

changingText.text = config[lvlCounter]['changingText']

changingText.style.cssText = config[lvlCounter]['changingTextStyle']

code.style.cssText = config[lvlCounter]['codeStyle']

leftArrow.bind('click', lvlBack)

rightArrow.bind('click', lvlAhead)

nextBtn.bind('click', lvlAhead)

Листинг 4. Скрипт для активации кнопки на JavaScript

let loadingScreen = document.querySelector('.loadingScreen')

let levels = document.querySelectorAll('.level')

let currentLvl = document.querySelector('.current')

let lvlBtn = document.querySelector(".next-btn")

let coddingField = document.querySelector(".coding")

let example = document.querySelector("#example")

let code = document.querySelector('#code');

let element = document.querySelector('#text');

let leftArrow = document.querySelector('.left-arrow');

let rightArrow = document.querySelector(".right-arrow");

function changeStyle() {

element.style.cssText = code.value;

}

function undisable(){

if(example.style.cssText == element.style.cssText){

lvlBtn.removeAttribute("disabled")

}

}

document.addEventListener('keydown', function(evt){

if(evt.key == "Enter" && example.style.cssText != element.style.cssText){

coddingField.classList.add('shake');

}

setTimeout(undisable, 500);

setTimeout(changeStyle, 1000);

})

function hideLoading(){

loadingScreen.setAttribute("hidden", "hidden")

}

setTimeout(hideLoading, 6000)

leftArrow.addEventListener("click", function(){

code.value = "";

lvlBtn.setAttribute("disabled", "disabled")

})

rightArrow.addEventListener("click", function(){

code.value = "";

lvlBtn.setAttribute("disabled", "disabled")

})

lvlBtn.addEventListener("click", function(){

levels[currentLvl.textContent-1].classList.add('level-passed')

code.value = "";

lvlBtn.setAttribute("disabled", "disabled")

code.focus()

})

Приложение 2

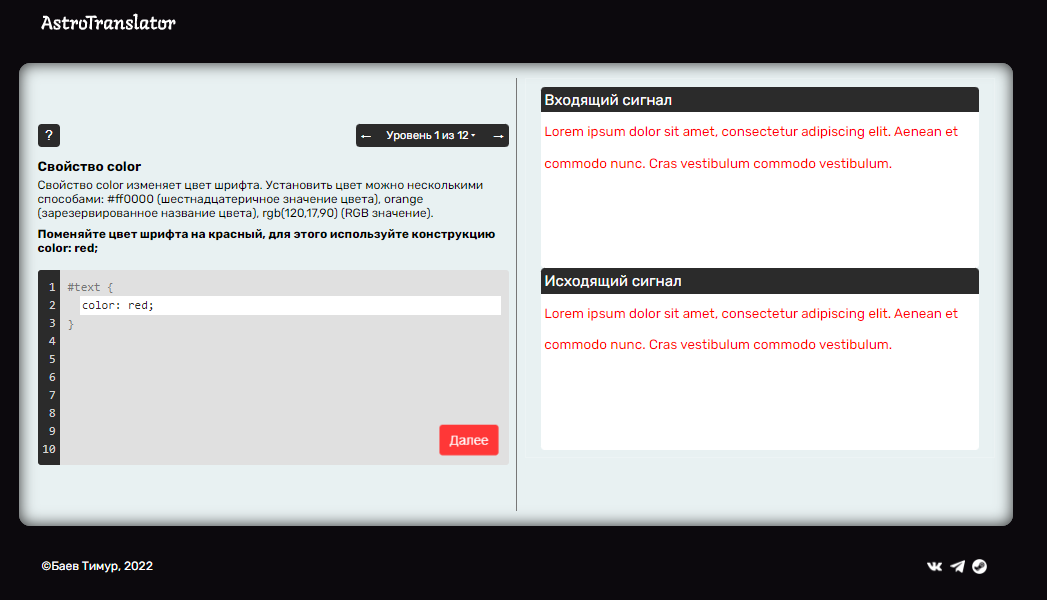


Рисунок 14. Демонстрация прохождения уровня

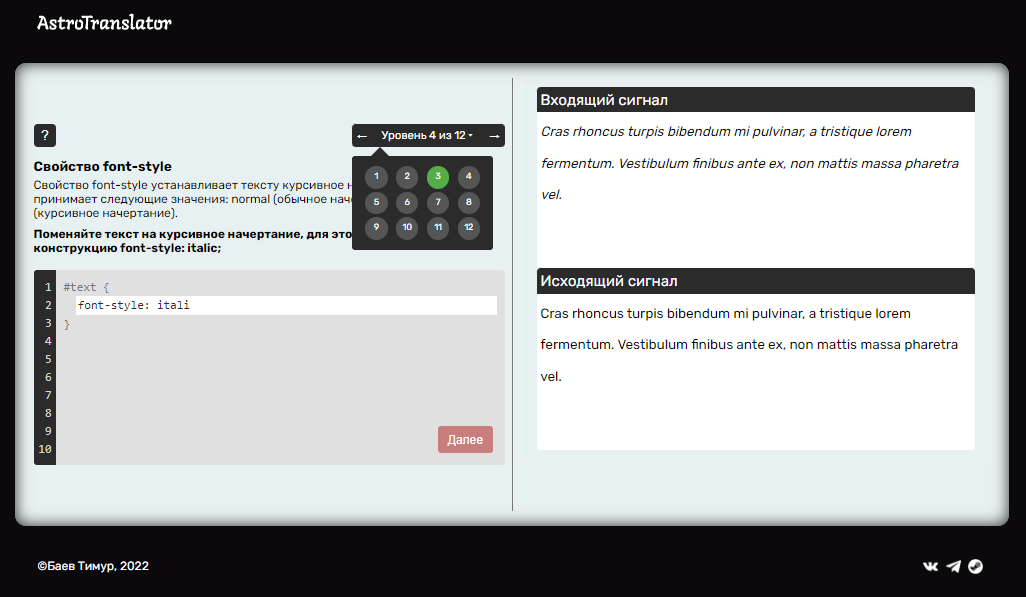


Рисунок 15. Демонстрация работы некоторых скриптов