**Содержание**

**Оглавление**

[**1.Постановка задачи** 2](#_Toc153874907)

[**1.1 Техническое задание** 2](#_Toc153874908)

[**1.2 Инструменты для создания** 2](#_Toc153874909)

[**2. Реализация структуры документа** 2](#_Toc153874910)

[**2.1 Структура HTML-документа** 2](#_Toc153874911)

[**2.2 Добавление таблиц стилей Sass и CSS** 4](#_Toc153874912)

[**2.3 Управление элементами DOM** 4](#_Toc153874913)

## **1.Постановка задачи**

### **1.1 Техническое задание**

Поставлена задача сверстать веб-сайт по макету. Выбор макета упал на данный пример:<https://www.figma.com/file/E7BcSTWqf7ltdAY3NQ6Ghp/space?type=design&node-id=0-1&mode=design&t=8PCcRtfTeIkB1Joj-0>. Проект должен быть свёрстан по данному примеру и быть точно таким же, либо же приближенным к нему. К данному проекту необходимо добавить какие-либо скрипты, которые будут выполнять логику на сайте. Проект будет иметь несколько анимации, а также будет иметь скрипт, который позволяет отправить текст, который пользователь заполнит в поле “input”, через телеграмм-бота мне в личные сообщения.

### **1.2 Инструменты для создания**

Для языка разметки будет использоваться язык разметки “HTML5”. Для того, чтобы проект стал похожим на макет, необходимо использовать стили. Для этого будет использоваться “CSS3”. Что бы выполнить анимации, а также отправку сообщения через “input” будет использован язык JavaScript.

# **2. Реализация структуры документа**

## **2.1 Структура HTML-документа**

В начале документа в теге <head></head> прописана служебная информация, название веб-сайта. В теге <title></title> прописано название сайта.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

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

<title>figmaMaket</title>

</head>

Листинг 2.1- содержимое шапки сайта

Далее расположен тег <body></body>, внутри которого размещаются все теги, в которых расположен сайт. Структура сайта располагается следующим образом: сайт состоит из 6-ти различных блоков, каждый из который отдельный и не зависит друг от друга. Каждый блок будет начинаться <div class="frame-1">, но вместо 1, будет номер блока.

<div class="counteiner-2">

<div class="frame-2"><div style="width: 679px;">

Get unlimited cloud storage space.

No more worrying.

</div>

<div class="frame-2L">

<div style="bottom: 100px;position: relative;left: -30%; "><input type="text" id="messageInput" name="" style="width: 300px; height: 50px;" class="input\_field" placeholder="Message"> </div>

<div style="bottom: 85px;position: relative; position: relative;left: -25%; "> <button class="btn" onclick="sendMessage()">Send</button></div>

</div>

</div>

</div>

Листинг 2.2- пример блока

## **2.2 Добавление таблиц стилей Sass и CSS**

Добавление таблиц стилей Sass (SCSS) и CSS позволяет упростить создание стилистического оформления страницы. При верстке было выбрано использовать CSS3.

\*{

margin: 0;

font-family: Arial;

}

.counteiner-1{

height: 78px;

flex-shrink: 0;

}

.frame-1{

width: 960px;

height: 31.8px;

top: 23px;

left: 240px;

padding: 0px, 0px, 0px, 10px;

position: relative;

display: inline-flex;

padding-left: 0px;

justify-content: flex-end;

align-items: center;

gap: 334px;

}

Листинг 2.3 – Отрывок CSS кода

## **2.3 Управление элементами DOM**

Для управления элементами DOM использовался язык программирования JavaScript. Одним из обращений к элементам был document.getElementById(), который считается удобным, потому что он выбирает элемент с заданным значением атрибута id.

В данном проекте с помощью JS была создана анимация поднимания из блока и опускание обратно в блок телефона. Кроме того, был создан скрипт, позволяющий при заполнении пользователем формы и клике на кнопку “Отправить”, отправить набранный им текст в личное сообщение в телеграмме через телеграмм-бота. Для реализации этого, было необходимо сайт перенести на сервер. При помощи Node js, сайт был перенесен на локальный сервер и находится по адресу: <http://localhost:8080/>. Первым делом, создался телеграмм-бот. После его создания, необходимо узнать уникальный токен бота, а также id чата пользователя с ботом. Далее через браузерную функцию телеграмма, мы подключаемся и записываем туда свои данные. После соединения и выполнения всех необходимых функций, реализована функция, которая будет выполнять проверку на отправку сообщения, и будет выводить в диалоговом окне, статус отправки сообщения.

<script>

function sendMessage() {

const message = document.getElementById('messageInput').value;

const token = \*\*\*\*\*\*\*\*\*';

const chatId = '\*\*\*\*\*\*\*';

const url = `https://api.telegram.org/bot${token}/sendMessage`;

const data = new URLSearchParams({

chat\_id: chatId,

text: message,

});

fetch(url, {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

},

body: data,

})

.then(response => response.json())

.then(result => {

if (result.ok) {

alert('Сообщение успешно отправлено в Telegram!');

} else {

alert('Ошибка при отправке сообщения в Telegram.');

}

})

.catch(error => {

console.error('Ошибка:', error);

alert('Произошла ошибка. Пожалуйста, попробуйте еще раз.');

});

}

</script>

Листинг 2.4 – Скрипт отправки сообщения