Министерство просвещения РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Глазовский государственный инженерно-педагогический
университет имени В. Г. Короленко»

Факультет информатики, физики и математики

Кафедра математики и информатики

Среднее профессиональное образование

ОТЧЕТ

по учебной практике

ПМ. 04 СОПРОВОЖДЕНИЕ И ОБСЛУЖИВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОМПЬЮТЕРНЫХ СИСТЕМ

Специальность 09.02.07 «Информационные системы и программирование»

Выполнила:
студентка 3 курса СИ131 группы
Долина Н.А.

Руководитель практики:
Касаткин К.А., старший
преподаватель

_____
оценка

_____
дата, подпись руководителя

г. Глазов 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В качестве программного проекта учебной практики была выбрана программа кодировки цветов и измерения их цветовых координат.

Назначение кодировки состоит в однозначном определении нужного оттенка. Шестнадцатеричная кодировка цветов используется при создании веб-страниц, разработке приложений и графического интерфейса пользователя. Цветовые координаты позволяют с точностью определить цвет любой точки цветовой модели. Цветовые координаты находят свое применение в работе с цифровой графикой.

Задачами работы являются анализ предметной области выбранного проекта и создание программного проекта в соответствии с составленным для него техническим заданием, а также изучение основ тестирования и документирования программного обеспечения, проектирования UML-диаграмм и работы с системой контроля версий Git.

1. Анализ предметной области

В соответствии с выбранным программным проектом был проведен анализ предметной области.

Назначением проектируемого программного средства является работа с цветовыми моделями.

Цветовая модель – это способ описания цвета в виде совокупности числовых параметров [7].

Цветовые модели обладают такими характеристиками, как битовая глубина цвета и цветовой охват.

Битовая глубина цвета – суммарное количество двоичных разрядов, используемых для представления информации о цвете одного пикселя [4].

Цветовой охват модели – это вся совокупность цветов, которые могут быть воспроизведены с использованием данной цветовой модели [4].

По принципу действия цветовые модели делятся на три класса [4]:

1. **Аддитивные**, основанные на сложении цветов (RGB);
2. **Субтрактивные**, основу которых составляет операция вычитания цветов (CMYK);
3. **Перцепционные** (интуитивные), базирующиеся на восприятии (HSB, HLS, Lab).

Проектируемая программа рассчитана на работу с такими цветовыми моделями, как RGB, CMYK, HSB и HLS.

Цветовая модель RGB используется при описании цветов, получаемых смешением световых лучей. Для записи кода используется десятичное или шестнадцатеричное представление кода [4].

Цветовая модель CMYK состоит из трех базовых цветов: Cyan, Magenta, Yellow [4].

Цветовая модель HSB основана на трех параметрах: H (hue) – оттенок, тон; S (saturation) – насыщенность и B (brightness) – яркость [4].

Цветовая модель HLS основана на 3 параметрах: H (hue) – оттенок, L (lightness) – освещенность и S (saturation) – насыщенность [4].

Перечисленные цветовые модели обладают такими параметрами, как цветовые тона, насыщенность и яркость.

Цветовой тон – свет с доминирующей длиной волны [4].

Насыщенность характеризует чистоту цвета [4].

Яркость понимается как степень освещенности [4].

2. Анализ существующих аналогов. Определение функционала

Были рассмотрены примеры программ, имеющих сходный функционал с проектируемой программой.

bColor – программа, предназначенная для представления выбранного цвета в цифровых форматах HTML, VB, C++, Delphi, RGB (hex), RGB (dec), DEC [5]. Интерфейс программы представлен на Рисунке 1 [2].

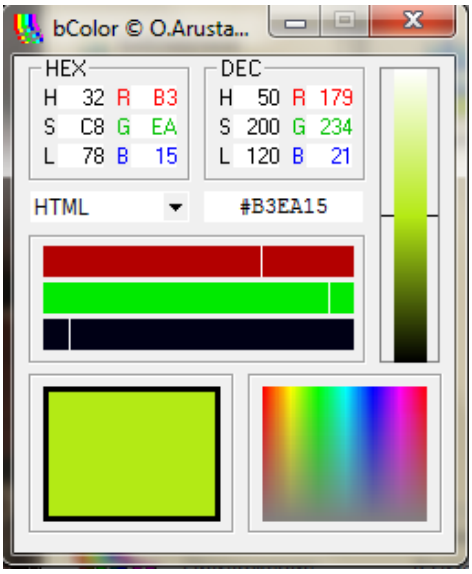


Рисунок 1 – Интерфейс программы bColor

Microsearch Color Picker измеряет цвета в моделях RGB и HSL, а также вычисляет их коды в шестнадцатеричном формате и HTML [5]. На Рисунке 2 [1] представлено окно приложения.

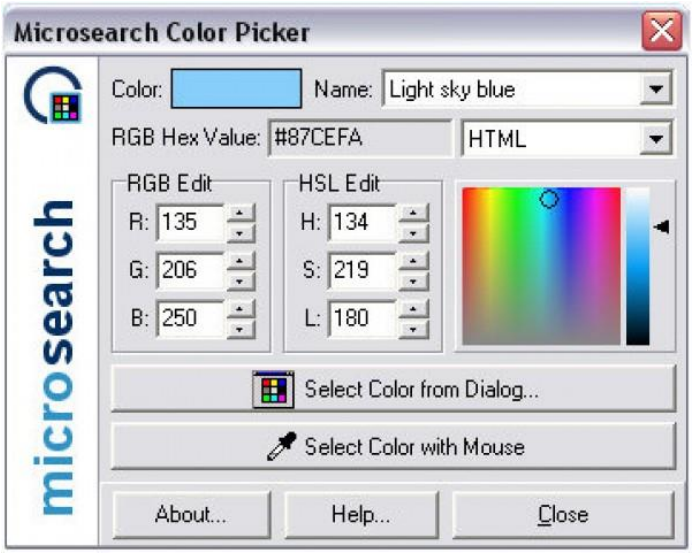


Рисунок 2 – Окно программы

HTML Colors – приложение для определения кода цвета в HTML и измерения его в формате RGB [5]. На Рисунке 3 [3] представлен интерфейс программы.

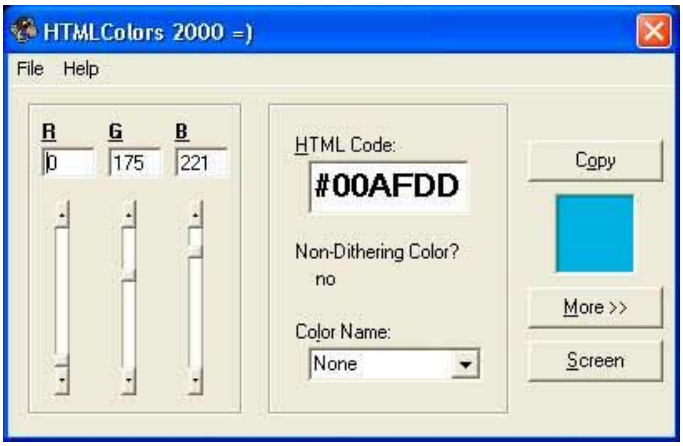


Рисунок 3 – Интерфейс приложения

Был определен ожидаемый функционал проектируемой программы: измерение выбранного цвета в моделях RGB, HSV, HSL и CMYK, а также вычисление его шестнадцатеричного кода в формате RGB.

Был разработан эскиз интерфейса приложения, представленный на Рисунке 4.

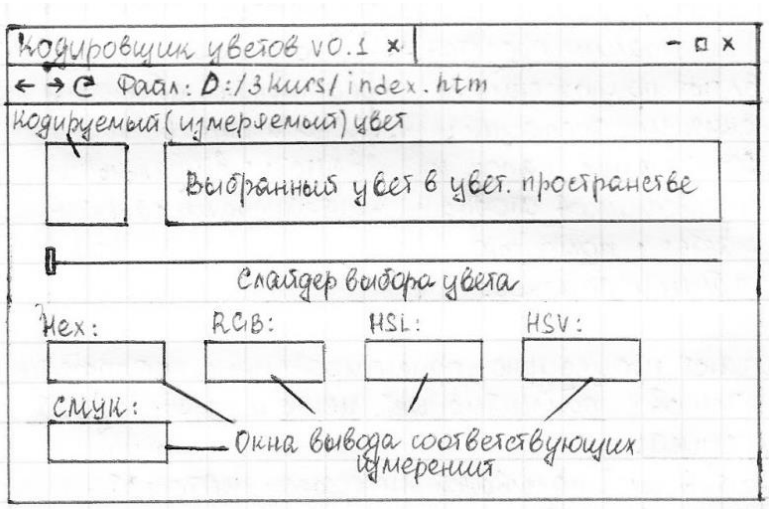


Рисунок 4 – Эскиз интерфейса программы

3. Анализ и выбор программных средств

В качестве программных средств реализации поставленной задачи были рассмотрены:

- Компилируемый статически типизированный язык программирования общего назначения C;
- Компилируемый статически типизированный язык программирования общего назначения C++, поддерживающий объектно-ориентированную разработку;
- Строго типизированный объектно-ориентированный язык программирования общего назначения Java;
- Мультипарадигменный язык программирования JavaScript;
- Стандартизированный язык разметки документов HTML;
- Формальный язык описания внешнего вида документа (веб-страницы) CSS;
- Интегрированная среда разработки Visual Studio, используемая для создания компьютерных программ, веб-сайтов, веб-приложений, веб-сервисов и мобильных приложений, поддерживающая разработку ПО на Visual C++, Visual C#, JavaScript;
- Кроссплатформенная свободная IDE для разработки на C, C++ и QML Qt Creator;
- Текстовый редактор Notepad++;
- Среда веб-разработки jsFiddle.

Для реализации поставленной задачи были выбраны такие программные средства, как мультипарадигменный язык программирования JavaScript, стандартизированный язык разметки документов HTML, формальный язык описания внешнего вида документа CSS и среда разработки jsFiddle.

Выбранный язык программирования используется для разработки веб-страниц и веб-приложений, которые в сравнении с локальными программами имеют такие преимущества, как простота доступа и развертывания. HTML и CSS используются как дополнение к языку программирования JavaScript с целью упрощения разметки и проектирования внешнего вида страницы. Выбранная среда разработки позволяет использовать при создании программного обеспечения выбранные программные средства.

4. Алгоритм, UML-диаграммы и блок-схема задачи

Для проектируемой программы была разработана UML-диаграмма пакетов. Диаграмма пакетов представлена на Рисунке 5.

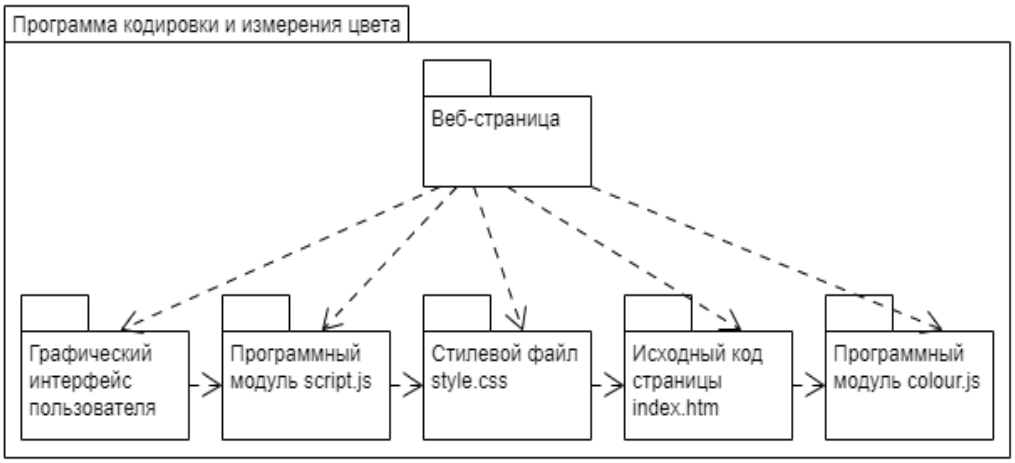


Рисунок 5 – Диаграмма пакетов приложения

Была разработана функциональная диаграмма проектируемой программы, представленная на Рисунке 6.

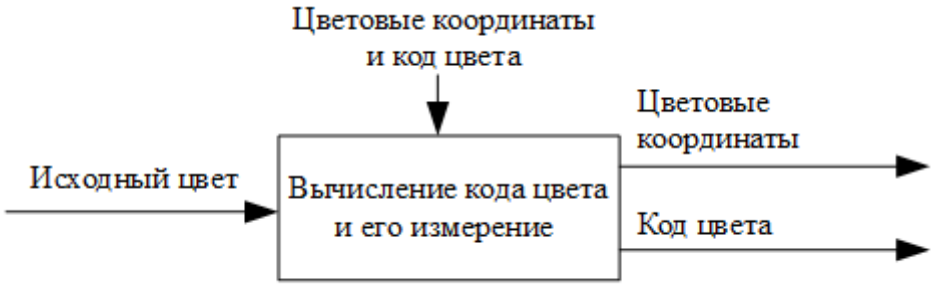


Рисунок 6 – Функциональная диаграмма

Была разработана диаграмма переходов состояний для программы, представленная на Рисунке 7.

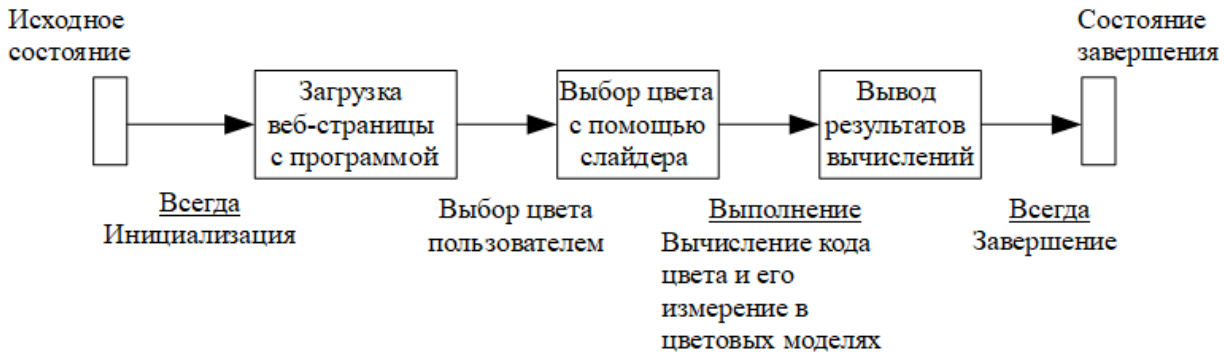


Рисунок 7 – Диаграмма переходов состояний

Была разработана диаграмма вариантов использования (представлена на Рисунке 8).

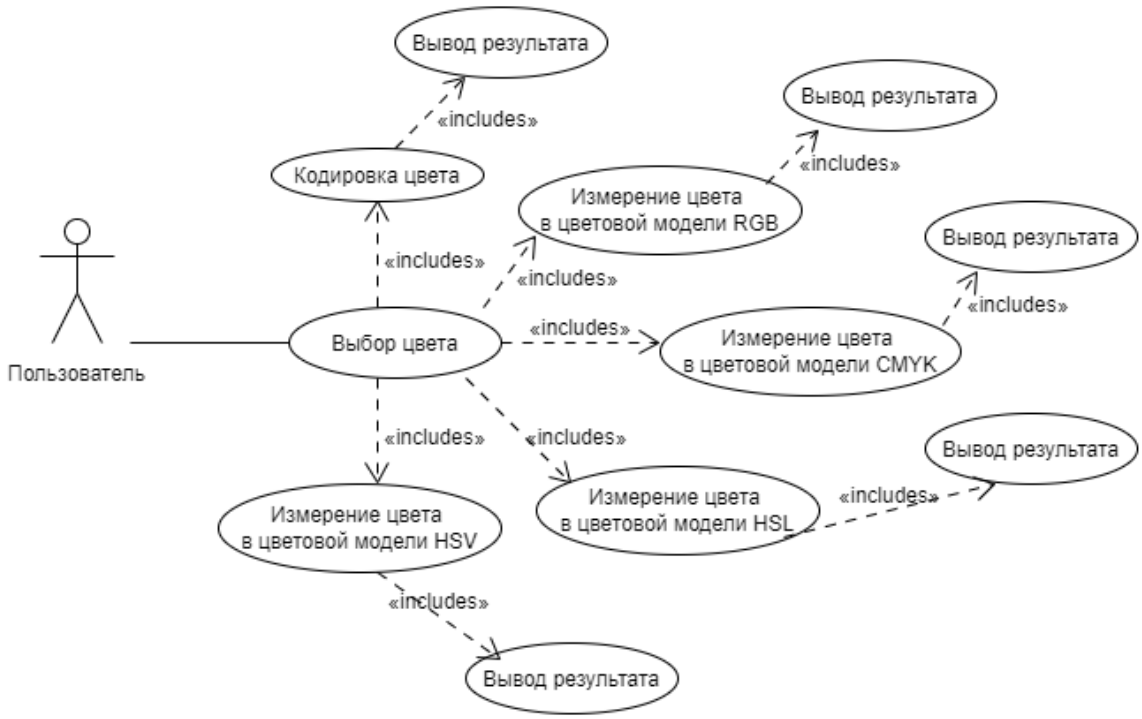


Рисунок 8 – Диаграмма вариантов использования

Была составлена диаграмма отношений компонентов данных (Рисунок 9).

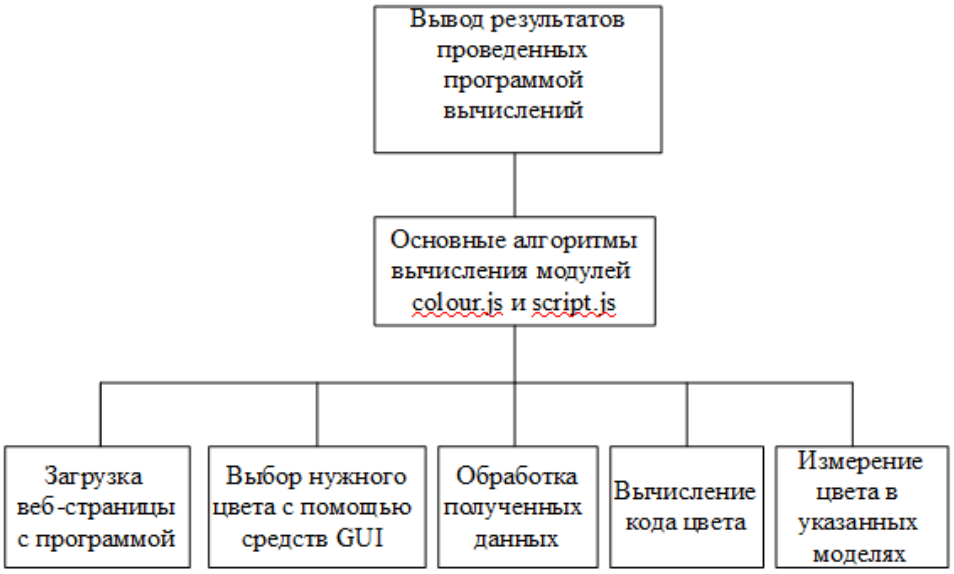


Рисунок 9 – Диаграмма отношений компонентов данных

Также были разработаны диаграммы классов, последовательностей и деятельности, представленные на Рисунках 10-12.

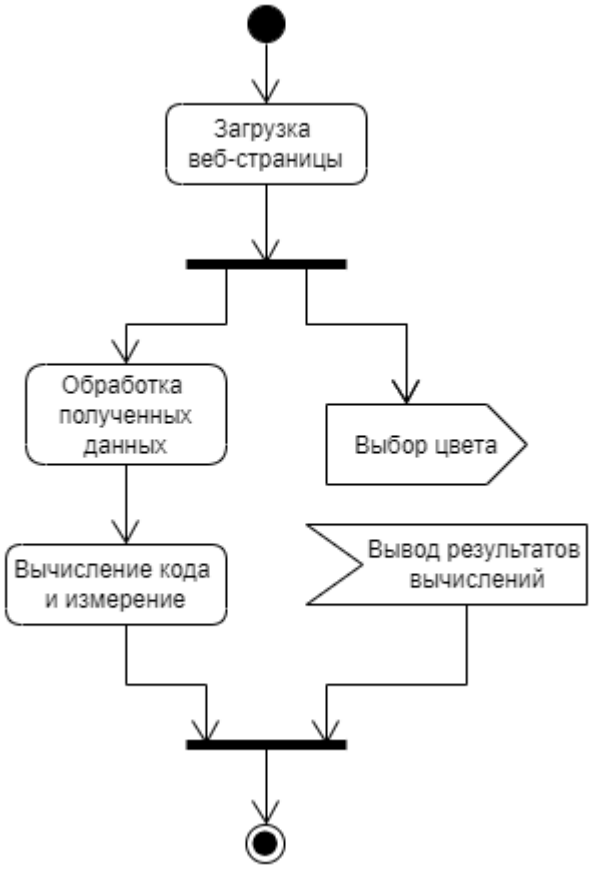


Рисунок 10 – Диаграмма деятельности

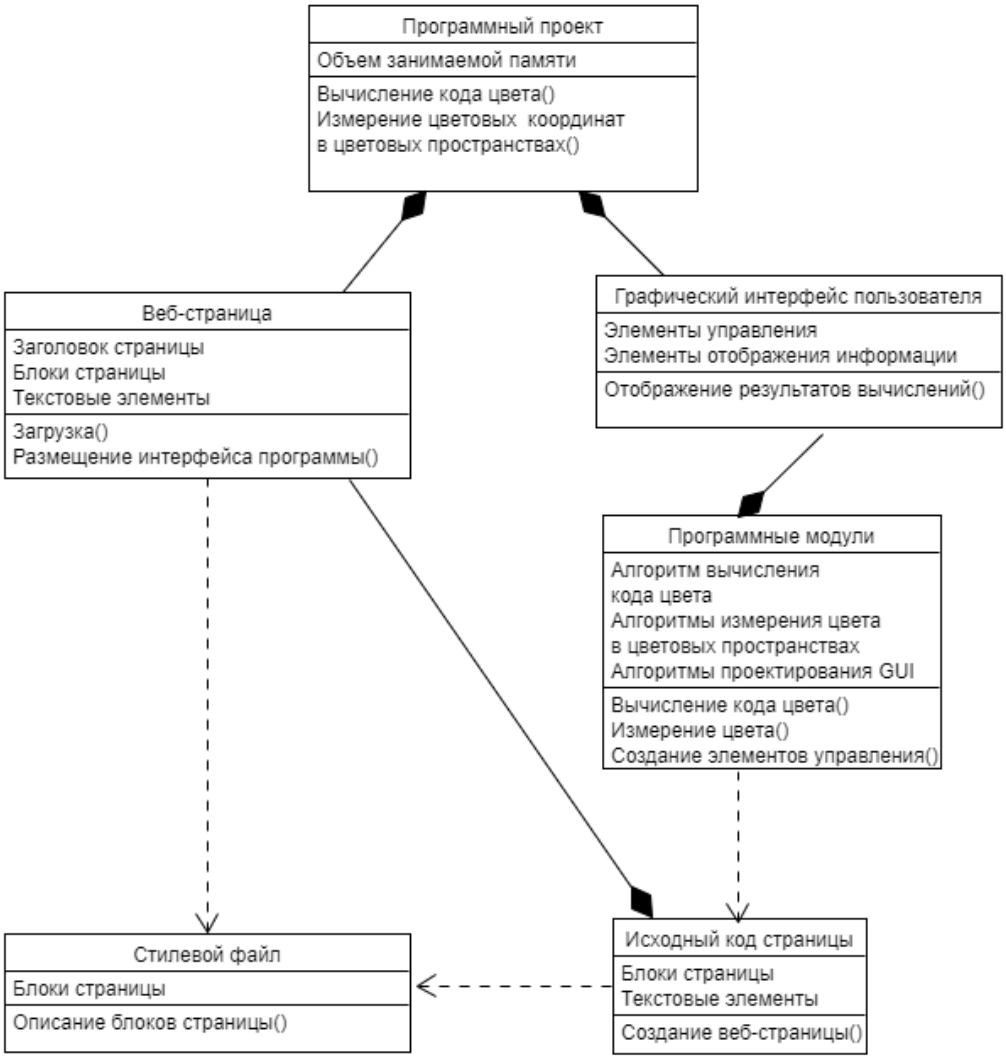


Рисунок 11 – Диаграмма классов

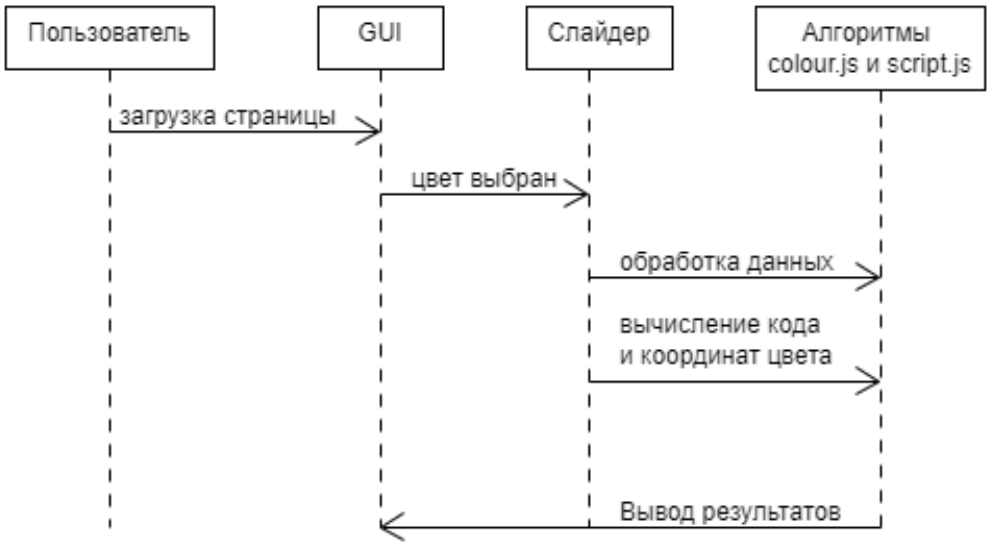


Рисунок 12 – Диаграмма последовательностей

Была составлена блок-схема алгоритма поставленной задачи (Рисунок 13).

Рисунок 13 – Блок-схема задачи

В соответствии с приведенными выше диаграммами и блок-схемой задачи был разработан алгоритм инкрементной разработки программы:

- 1 инкремент: начальная разметка веб-страницы и подключение к ней программного модуля JavaScript и стилевого файла;
- 2 инкремент: добавление на веб-страницу блоков для элементов графического интерфейса пользователя; полей для вывода вычислений кода и координат цвета;
- 3 инкремент: добавление описания стилей блоков, добавленных на страницу;
- 4 инкремент: создание функции измерения цвета в модели RGB в программном модуле;

- 5 инкремент: создание функции измерения цвета в HSB-модели;
- 6 инкремент: добавление в стилевой файл описания блоков слайдера, диапазона яркости цвета и окна для его отображения;
- 7 инкремент: создание в программном модуле функции измерения цвета в модели HLS;
- 8 инкремент: создание в программном модуле функции измерения цвета в CMYK;
- 9 инкремент: создание в программном модуле функции вычисления шестнадцатеричного кода цвета;
- 10 инкремент: добавление в структуру проекта файла библиотеки jQuery;
- 11 инкремент: создание программного модуля для использования добавленной библиотеки и разработка в нем функции связывания элементов графического элемента пользователя с алгоритмами вычисления шестнадцатеричного кода цвета и его координат;
- 12 инкремент: финальный; создание с помощью средств библиотеки jQuery функций обновления положения движка на слайдере и обновления выбираемого пользователем цвета.

5.Разработка

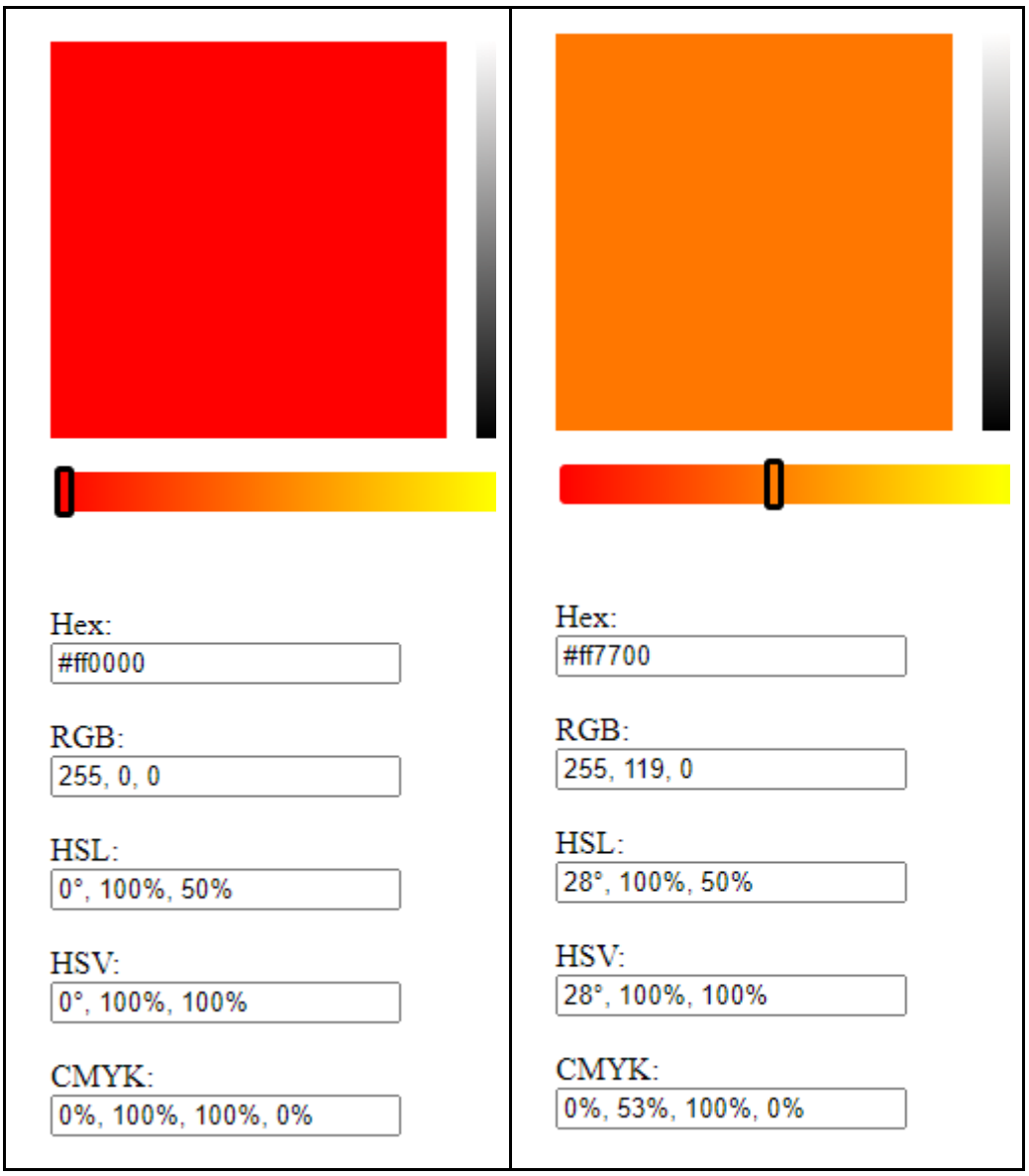
В соответствии с составленным алгоритмом инкрементной разработки было создано 12 инкрементов программы, которые представлены в Приложении 1

6.Тестирование

Было проведено функциональное тестирование программы, в ходе которого было выявлено, что программа соответствует требованиям технического задания. В Таблице 6.1 представлены результаты тестирования.

Таблица 6.1 Результаты функционального тестирования

Начальные значения	**Значения после изменения цвета**



7. Система контроля версий

Разработанные инкременты программы были помещены в репозиторий Git на веб-сервис GitHub.

Репозиторий Git – каталог файловой системы, в котором находятся файлы конфигурации, файлы журналов операций, выполняемых над репозиторием, индекс расположения файлов и хранилище, содержащее контролируемые файлы [6].

Один из видов объектов, помещаемых в хранилище Git – **фиксация** (**коммит**). Эти объекты хранят метаданные для каждого изменения, произошедшего в репозитории, включая имя автора, дату фиксации и сообщение журнала [6].

Каждой фиксации назначается имя с помощью другого вида объекта хранилища Git – **тега** [6].

Ниже представлен процесс создания репозитория для каталога с разработанными инкрементами.

На Рисунке 10 представлена команда создания пустого репозитория в каталоге с инкрементами:

```
nat@nat-Inspiron-15-3552:~$ cd Рабочий\ стол/
nat@nat-Inspiron-15-3552:~/Рабочий стол$ ls -l
итого 36
-rw-rw-r--  1 nat nat 20248 мая 25 13:39 1.png
drwxrwxr-x  2 nat nat  4096 мая 24 18:14 GIMP
drwxrwxr-x 14 nat nat  4096 мая 25 14:28 PM01
drwxrwxr-x  7 nat nat  4096 апр 13 12:36 Practice
drwxrwxr-x  4 nat nat  4096 мая 10 12:50 Курсовая
nat@nat-Inspiron-15-3552:~/Рабочий стол$ cd PM01
nat@nat-Inspiron-15-3552:~/Рабочий стол/PM01$ git init
Инициализирован пустой репозиторий Git в /home/nat/Рабочий стол/PM01/.git/
nat@nat-Inspiron-15-3552:~/Рабочий стол/PM01$
```

Рисунок 31 – Создание пустого репозитория

На Рисунке 11 представлена команда просмотра состояния рабочего дерева Git.

```
nat@nat-Inspiron-15-3552:~/Рабочий стол$ cd PM01
nat@nat-Inspiron-15-3552:~/Рабочий стол/PM01$ git status
На ветке master

Еще нет коммитов

Изменения, которые будут включены в коммит:
  (используйте «git rm --cached <файл>…», чтобы убрать из индекса)
        новый файл:     "1 \320\270\320\275\320\272\321\200\320\265\
        новый файл:     "1 \320\270\320\275\320\272\321\200\320\265\
        новый файл:     "1 \320\270\320\275\320\272\321\200\320\265\
        новый файл:     "10 \320\270\320\275\320\272\321\200\320\265
        новый файл:     "10 \320\270\320\275\320\272\321\200\320\265
        новый файл:     "10 \320\270\320\275\320\272\321\200\320\265
```

Рисунок 32 – Состояние рабочего дерева

Ниже представлена команда создания коммита для файлов 2 инкремента (Рисунок 33).

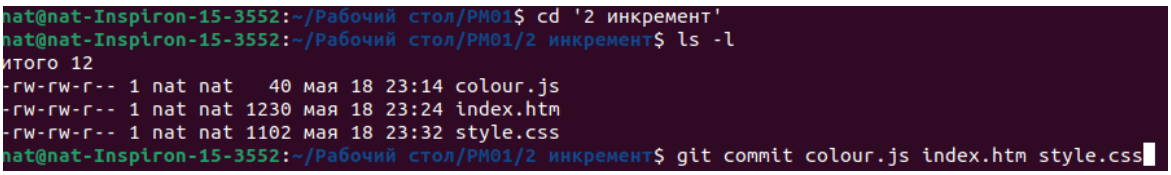


Рисунок 33 – Создание фиксации

На Рисунке 34 показано состояние рабочего дерева после создания фиксаций для каждого инкремента.

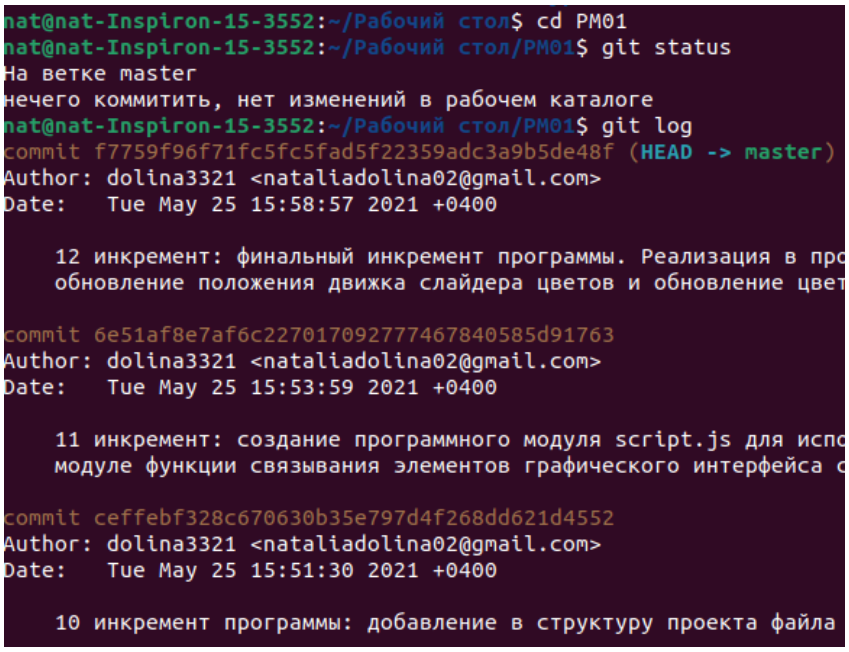


Рисунок 34 – Состояние рабочего дерева

На Рисунках 35-37 представлен процесс создания удаленного репозитория и загрузки фиксаций в пустой репозиторий на веб-сервисе GitHub.

```
nat@nat-Inspiron-15-3552:~/Рабочий стол/PM01$ curl -u 'dolina3321' https://github.com/dolina3321/-01' {"name":"Учебная практика ПМ01 25.05.202
1"}'
Enter host password for user 'dolina3321':

<html>
  <head>
    <meta content="origin" name="referrer">
    <title>Bad request &middot; GitHub</title>
    <meta name="viewport" content="width=device-width">
    <style type="text/css" media="screen">
      body {
        background-color: #f6f8fa;
        color: rgba(0, 0, 0, 0.5);
        font-family: -apple-system,BlinkMacSystemFont,Segoe UI,Helvetica,Arial,sans-serif,Apple Color Emoji,Segoe UI Emoji,Segoe UI Symbol;
        font-size: 14px;
        line-height: 1.5;
      }
      .c { margin: 50px auto; max-width: 600px; text-align: center; padding: 0 24px; }
      a { text-decoration: none; }
      a:hover { text-decoration: underline; }
      h1 { color: #24292e; line-height: 60px; font-size: 48px; font-weight: 300; margin: 0px; }
      p { margin: 20px 0 40px; }
      #s { margin-top: 35px; }
      #s a {
        color: #666666;
        font-weight: 200;
        font-size: 14px;
        margin: 0 10px;
```

Рисунок 35 – Создание удаленного репозитория

```
nat@nat-Inspiron-15-3552:~/Рабочий стол/PM01$ git remote add origin https://github.com/dolina3321/-01
nat@nat-Inspiron-15-3552:~/Рабочий стол/PM01$ git push -u origin master
Username for 'https://github.com': dolina3321
Password for 'https://dolina3321@github.com':
Перечисление объектов: 55, готово.
Подсчет объектов: 100% (55/55), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (52/52), готово.
Запись объектов: 100% (55/55), 39.42 KiB | 1.71 MiB/s, готово.
Total 55 (delta 26), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (26/26), done.
To https://github.com/dolina3321/-01
 * [new branch]      master -> master
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
nat@nat-Inspiron-15-3552:~/Рабочий стол/PM01$
```

Рисунок 36 – Загрузка фиксаций в пустой репозиторий

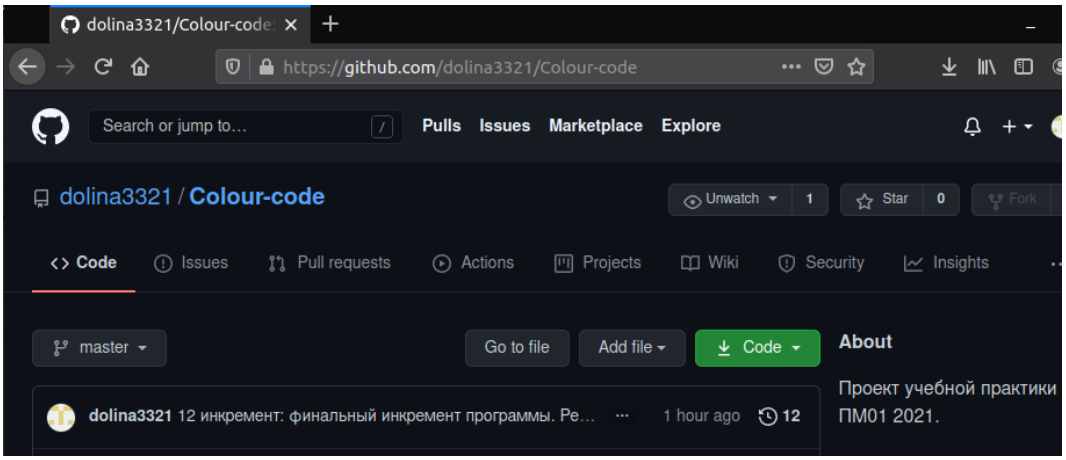


Рисунок 37 – Репозиторий на веб-сервисе GitHub

Также был создан репозиторий для программной документации проекта. Скриншот приведен на Рисунке 38.

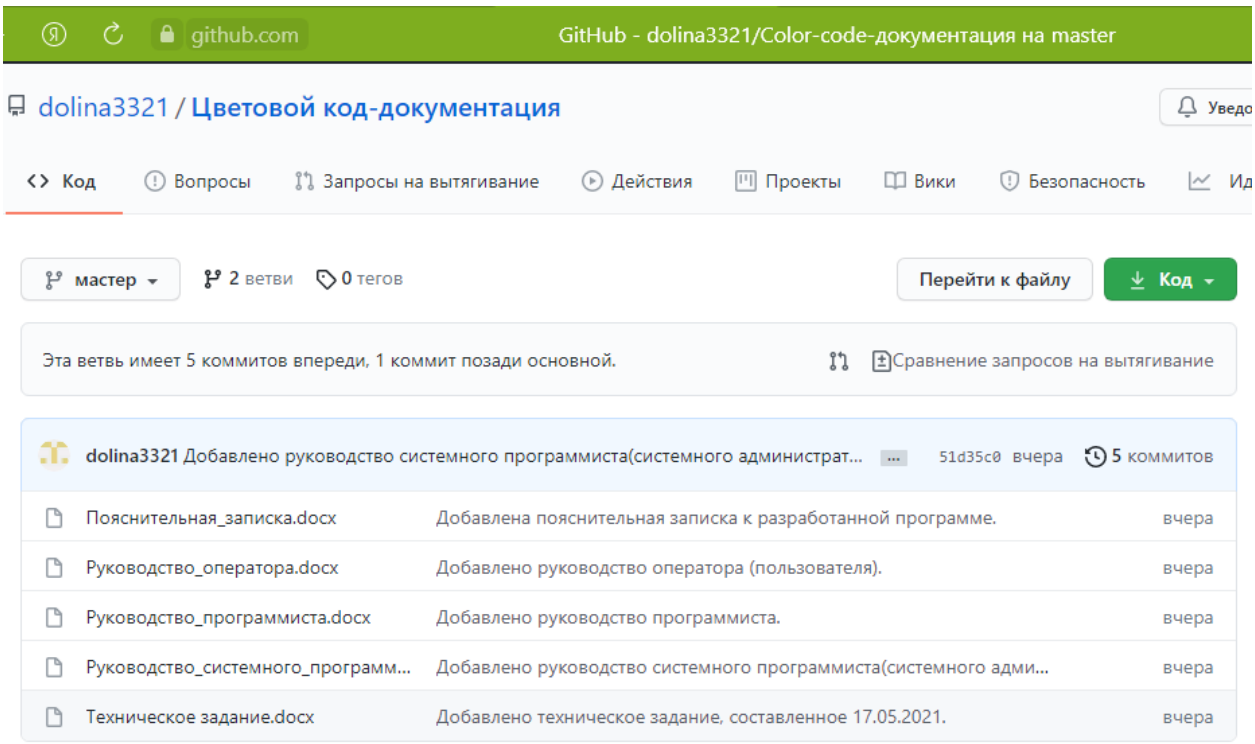


Рисунок 38 – Репозиторий с программной документацией

ЗАКЛЮЧЕНИЕ

Таким образом, в ходе работы были достигнуты цель и задачи практики – было разработано прикладное программное обеспечение по выбранной теме, изучены основы тестирования программного обеспечения и работы с системой контроля версий.

Также в заключение можно сделать вывод о дальнейшем совершенствовании программы, которое может состоять в реализации возможности ввода координат и кода цвета пользователем с клавиатуры и измерения кода цвета в формате HTML.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. http://bestsoft.ru/files/3/3116/screen/big/Microsearch_Color_Picker.jpg
2. http://cdn.bolshoyvopros.ru/files/users/images/b6/b9/b6b9d00b3b36564b47a8abd75511e179.png
3. https://rugraphics.ru/sites/default/files/Programms/Editimage/html_colors_2000.jpg
4. Кодирование цвета. Цветовые модели и цветовые режимы. URL: https://life-prog.ru/2_109985_kodirovanie-tsveta-tsvetovie-modeli-i-tsvetovie-rezhimi.html (дата обращения: 17.05.2021). – Режим доступа: свободный.
5. Программы для определения цвета. URL: https://rugraphics.ru/forimage/programmy-dlya-opredeleniya-tsveta (дата обращения: 17.05.2021). – Режим доступа: свободный.
6. Фишерман, Л.В. Git. Практическое руководство. Управление и контроль версий в разработке программного обеспечения – СПб.: Наука и Техника, 2021. – 304 с., ил. с.44-45. – ISBN 978-5-94387-547-2
7. Цветовая модель. URL: https://ru.wikipedia.org/wiki/Цветовая_модель#:~:text=Цветовая%20модель%20%20математическая%20модель,задаваемые%20моделью%2C%20определяют%20цветовое%20пространство (дата обращения: 17.05.2021). – Режим доступа: свободный.

ПРИЛОЖЕНИЯ

Приложение 1. Инкременты разработки программы

1 инкремент

Создание начальной версии веб-страницы, программного модуля и стилевого файла. На Рисунке 14-15 представлены скриншоты среды разработки.

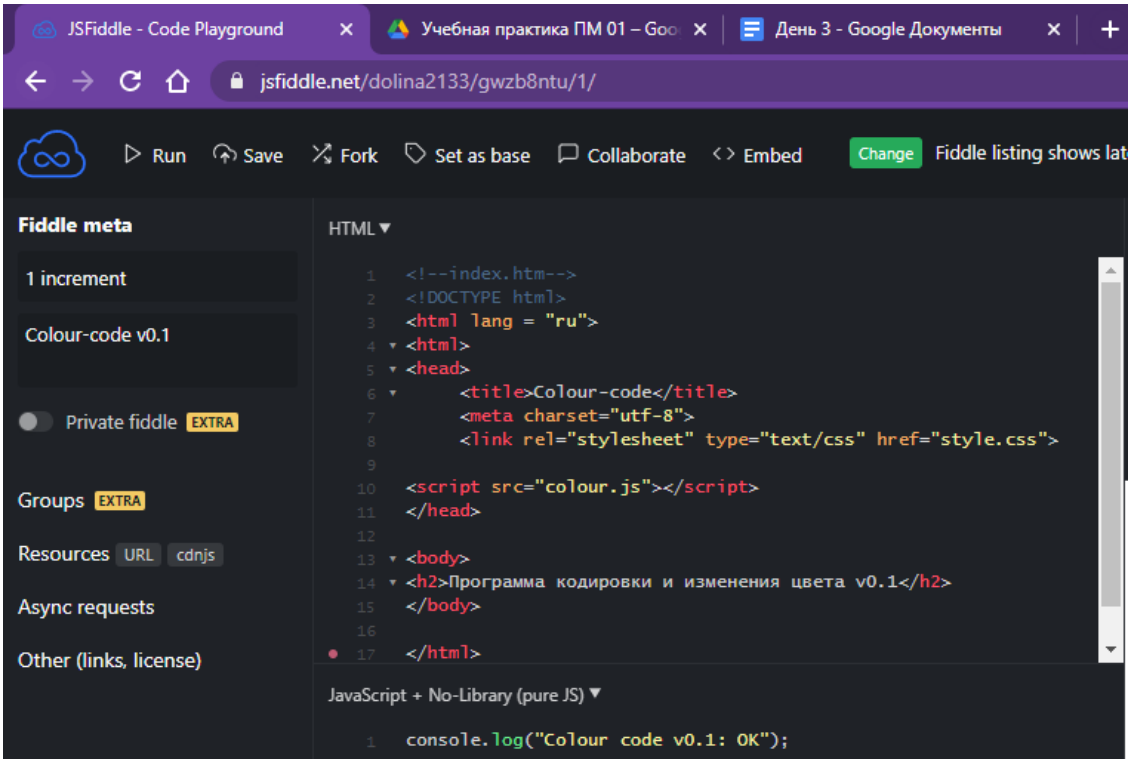


Рисунок 14 – Исходный код страницы и программный модуль в среде разработки

Рисунок 15 – Стилевой файл

На Рисунке 16 показана веб-страница.

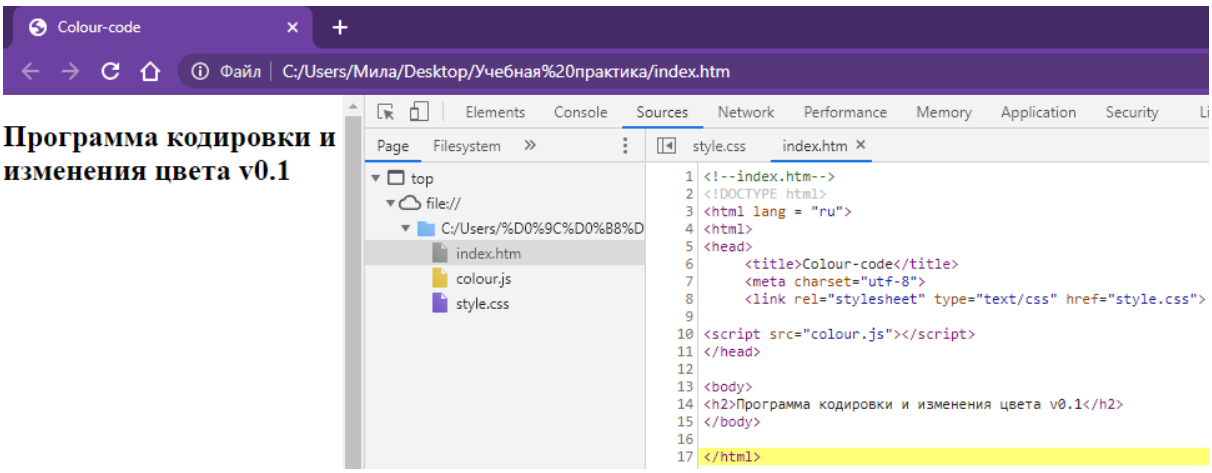


Рисунок 16 – Веб-страница

2 инкремент

В исходный код страницы добавлены блоки элементов графического интерфейса пользователя: блок слайдера цветов и полей вывода вычисляемых измерений с начальными значениями. Скриншот среды разработки приведен на Рисунке 17.

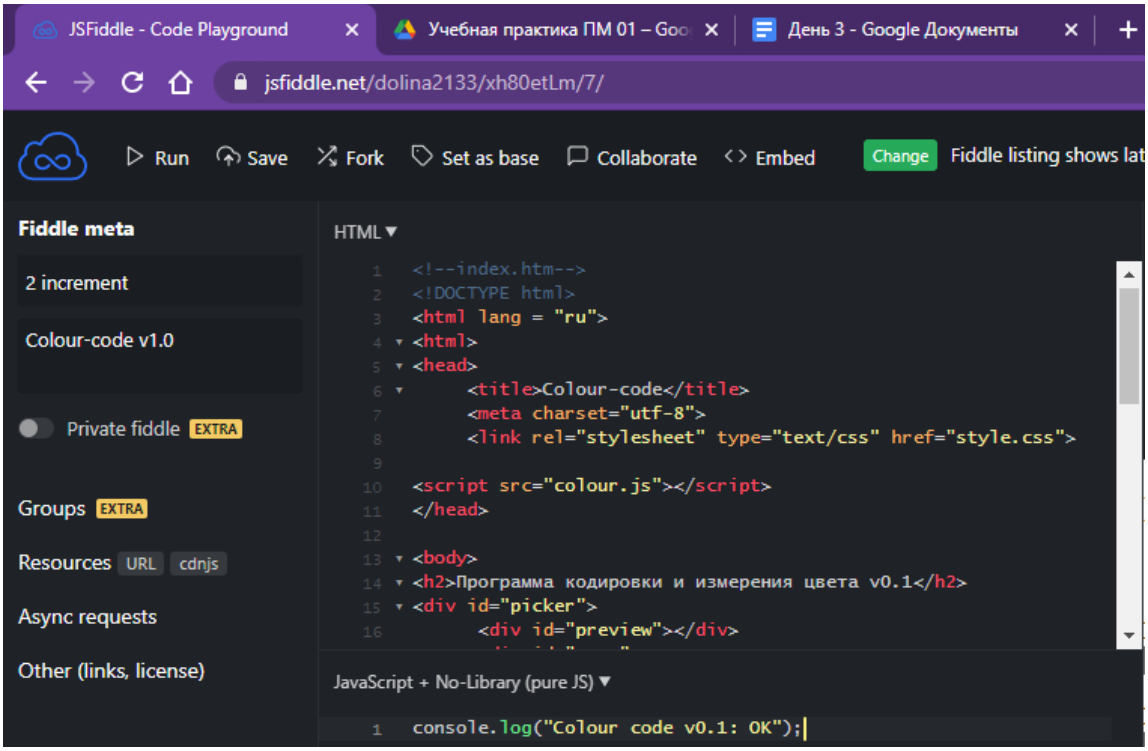


Рисунок 17 – Исходный код страницы и программный модуль

Ниже представлен исходный код страницы.

//index.htm

```
<div id="picker">
      <div id="preview"></div>
      <div id="pane">
        <div id="white-to-pure-color"></div>
        <div id="transparent-to-black"></div>
        <div id="knob"></div>
      </div>

      <input id="hue" type="range" min="0" max="360" value="0">

      <p id="formats">
        Hex:<br>
        <input type="text" value="#ff0000" id="hex"
class="color-format" /><br><br>
        RGB:<br>
        <input type="text" value="255, 0, 0" id="rgb"
class="color-format" /><br><br>
        HSL:<br>
        <input type="text" value="0°, 100%, 50%" id="hsl"
class="color-format" /><br><br>
        HSV:<br>
        <input type="text" value="0°, 100%, 100%" id="hsv"
class="color-format" /><br><br>
        CMYK:<br>
        <input type="text" value="0%, 100%, 100%, 0%" id="cmyk"
class="color-format" /><br><br>
      </p>
    </div>
</body>

</html>
```

3 инкремент

Описание стилей добавленных на страницу блоков в стилевом файле style.css. Скриншоты среды разработки и веб-страница представлены на Рисунках 18-19.

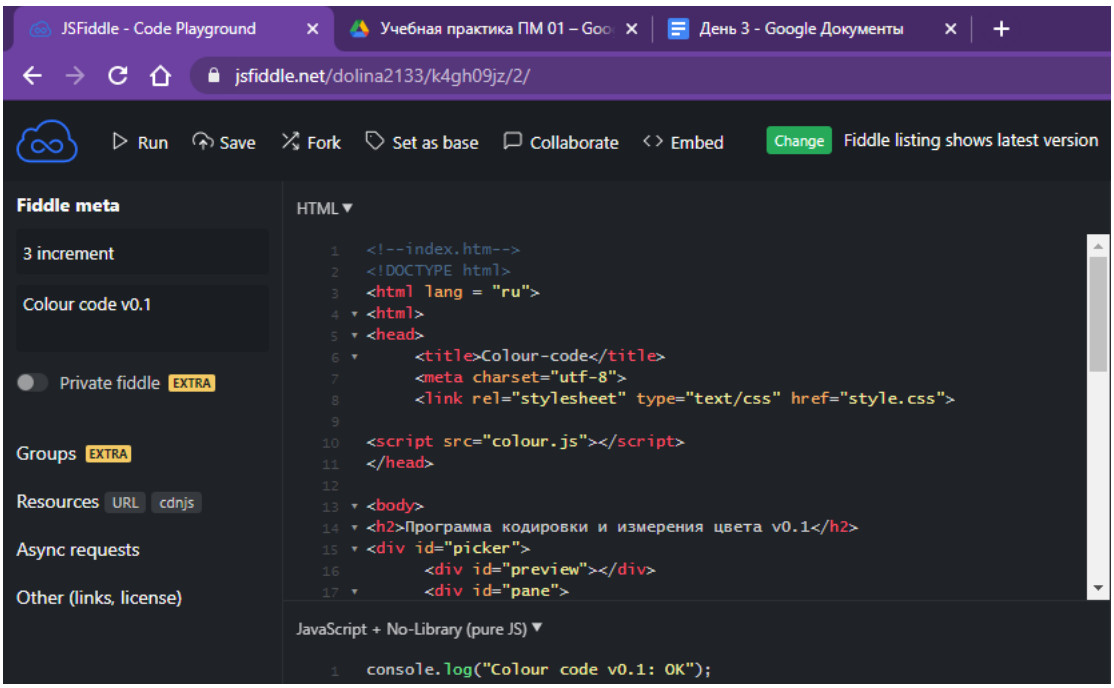


Рисунок 18 – 3 инкремент в среде разработки

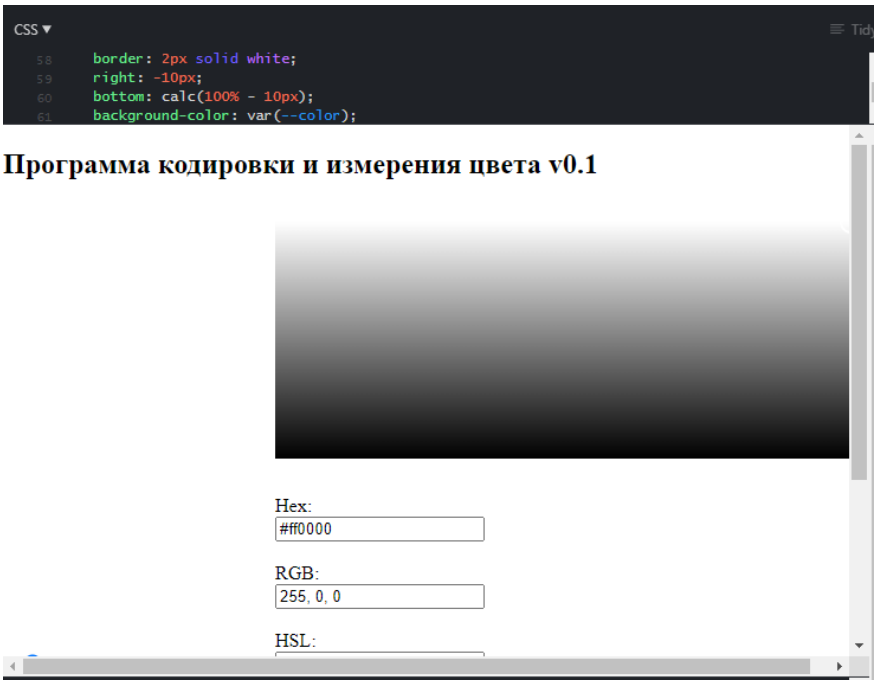


Рисунок 19 – Фрагмент стилевого файла и веб-страница

Ниже представлен код стилевого файла.

//style.css

```
* {
  box-sizing: border-box;
}
```

```
html, body {
  margin: 0;
  width: 100vw;
  min-height: 100vh;
}
#picker {
  display: grid;
  padding: 15px;
  grid-template-columns: auto 1fr;
  grid-template-areas:
    'preview pane'
    'hue hue'
    'opacity opacity'
    'formats formats';
  grid-gap: 15px;
  justify-items: stretch;
}
#preview {
  min-width: 200px;
  min-height: 200px;
  background-color: var(--color);
  grid-area: preview;
}
#pane {
  min-width: 200px;
  min-height: 200px;
  position: relative;
  cursor: pointer;
  grid-area: pane;
}
#white-to-pure-color {
  width: 100%;
  height: 100%;
  position: absolute;
  background-image: linear-gradient(to right, white, hsl(var(--
hue), 100%, 50%));
}
#transparent-to-black {
  width: 100%;
  height: 100%;
  position: absolute;
  background-image: linear-gradient(transparent, black);
}

#knob {
  position: absolute;
  width: 20px;
  height: 20px;
  border-radius: 50%;
  border: 2px solid white;
  right: -10px;
  bottom: calc(100% - 10px);
```

```
  background-color: var(--color);
}
```

4 инкремент

В программный модуль добавлена функция измерения выбранного пользователем цвета в модели RGB. На Рисунке 20 представлен скриншот среды разработки.

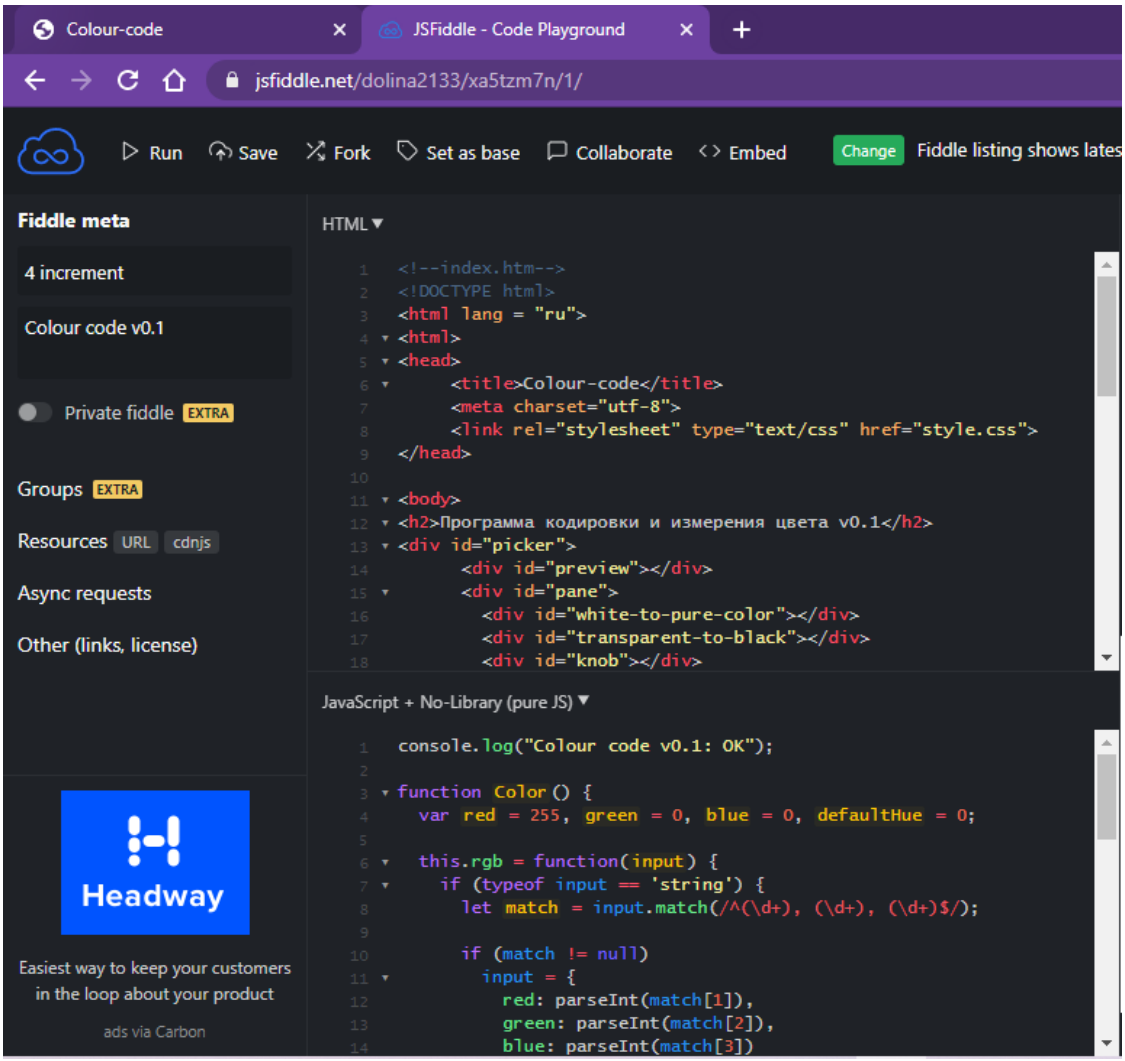


Рисунок 20 – Инкремент в среде разработки

//colour.js

```
console.log("Colour code v0.1: OK");

function Color() {
  var red = 255, green = 0, blue = 0, defaultHue = 0;
```

```
  this.rgb = function(input) {
    if (typeof input == 'string') {
      let match = input.match(/^(\d+), (\d+), (\d+)$/);

      if (match != null)
        input = {
          red: parseInt(match[1]),
          green: parseInt(match[2]),
          blue: parseInt(match[3])
        };
    }

    if (typeof input == 'object') {
      let isValid = (a) => typeof a == 'number' && a >= 0 && a
<= 255,
          validObj = true;

      if (isValid(input.red)) {validObj = true; red =
input.red;}
      if (isValid(input.green)) {validObj = true; green =
input.green;}
      if (isValid(input.blue)) {validObj = true; blue =
input.blue;}
      defaultHue = this.hsv().hue;
      if (validObj) return this;
    }

    return {
      red: red,
      green: green,
      blue: blue,
      toString: () => Math.round(red) + ', ' + Math.round(green)
+ ', ' + Math.round(blue)
    }
  }
}
```

5 инкремент

В программный модуль добавлена функция вычисления координат цвета в модели HSV. Ниже представлены скриншот разработки (Рисунок 21) и код программного модуля.

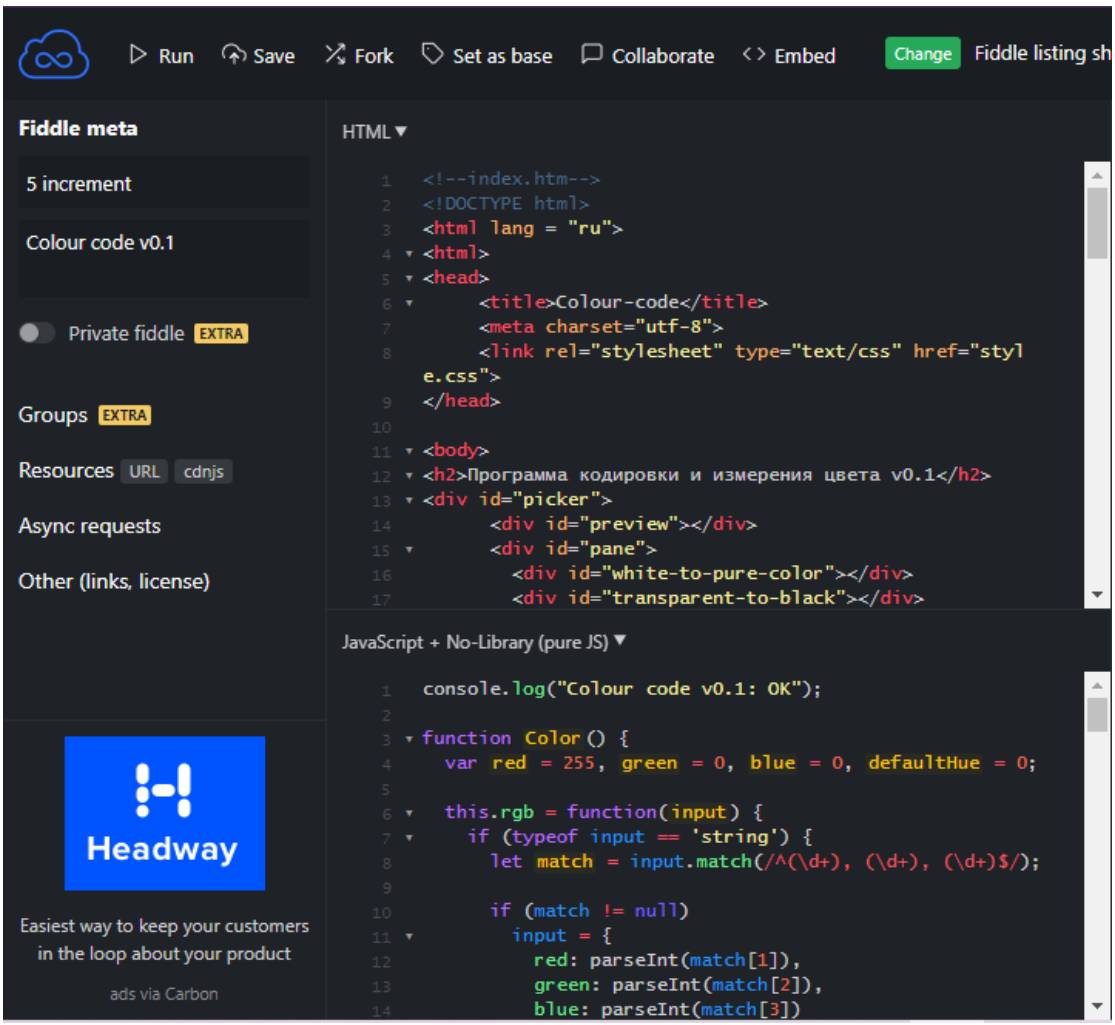


Рисунок 21 - 5 инкремент в среде разработки

```
//colour.js

// HSV color model (hue, saturation, value)
  this.hsv = function(input) {
    if (typeof input == 'string') {
      let match = input.match(/^(\d+)°, (\d+)%, (\d+)%$/);

      if (match != null)
        input = {
          hue: parseInt(match[1]),
          saturation: parseInt(match[2]),
          value: parseInt(match[3])
        };
    }

    if (typeof input == 'object') {

      var defaults = this.hsv(),
          valid = [['hue', 360], ['saturation', 100], ['value',
100]]
```

```
            .map(([prop, maxVal]) => {
              let validProp = typeof input[prop] == 'number' &&
input[prop] >= 0 && input[prop] <= maxVal;
              if (!validProp) input[prop] = defaults[prop];
              return validProp;
            })
            .reduce((a,b) => a || b);

      if (valid) {
        let s = input.saturation / 100,
            v = input.value / 100,
            h = input.hue / 60,
            c = v * s,
            x = c * (1 - Math.abs(h % 2 - 1)),
            a =
[[c,x,0],[c,x,0],[x,c,0],[0,c,x],[0,x,c],[x,0,c],[c,0,x]],
            m = v - c,
            [red, green, blue] = a[Math.ceil(h)].map(v => 255 *
(v + m));


        this.rgb({red: red, green: green, blue: blue});
        defaultHue = input.hue;
        return this;
      }
    }

    var rgb = this.rgb(),
        r = rgb.red / 255,
        g = rgb.green / 255,
        b = rgb.blue / 255,
        value = Math.max(r, g, b),
        c = value - Math.min(r, g, b);

    if (c == 0) var hue = defaultHue;
    else if (value == r) var hue = (g < b) ? 360 + 60 * (g - b)
/ c : 60 * (g - b) / c;
    else if (value == g) var hue = 60 * (2 + (b - r) / c);
    else var hue = 60 * (4 + (r - g) / c);

    var saturation = (value == 0) ? 0 : c / value * 100;

    return {
      hue: hue,
      saturation: saturation,
      value: value * 100,
      toString: () => Math.round(hue) + '°, ' +
Math.round(saturation) + '%, ' + Math.round(value * 100) + '%'
    };
}
}
```

6 инкремент

В стилевой файл добавлено описание линейки слайдера и окна отображения выбранного цвета. Скриншот среды разработки (Рисунок 22), веб-страницы (Рисунок 23) и код стилевого файла представлены ниже.

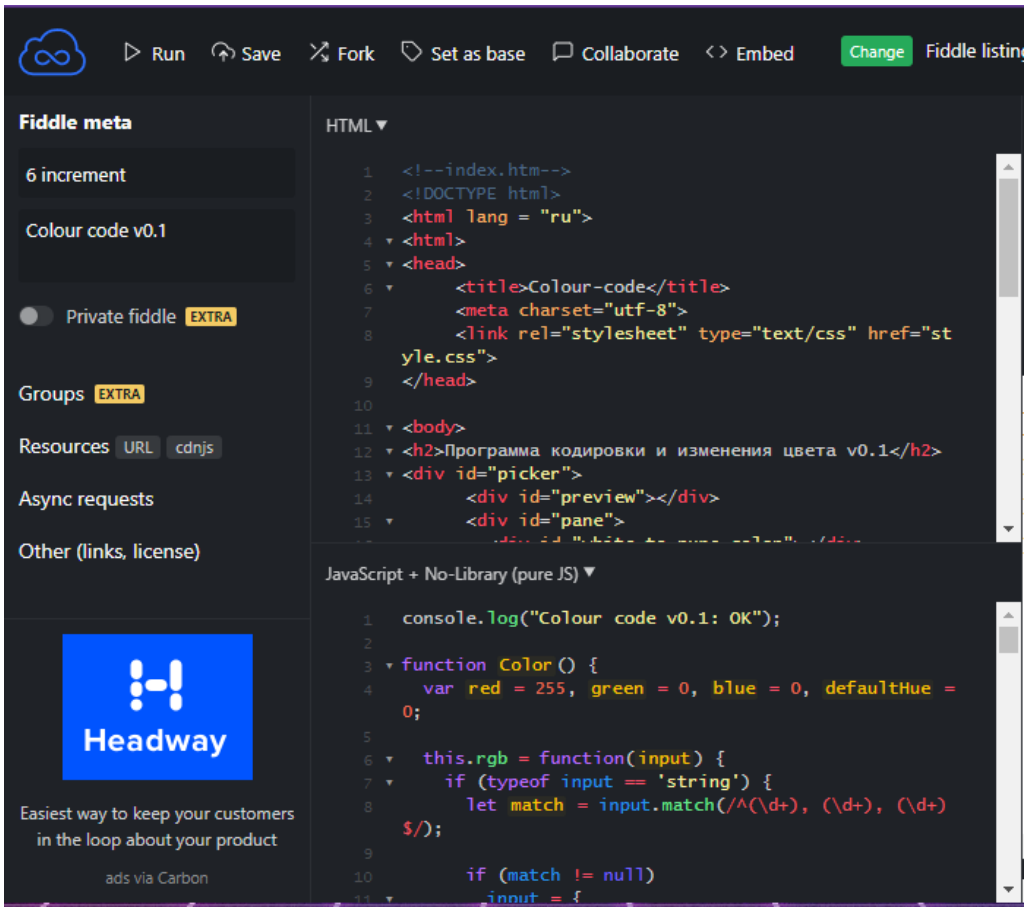


Рисунок 22 – 6 инкремент

//style.css

```
#hue {
  -webkit-appearance: none;
  height: 20px;
  outline: none;
  border-radius: 3px;
  background-image: linear-gradient(to right, red, yellow, lime, aqua, blue, fuchsia, red);
  cursor: pointer;
  grid-area: hue;
}
```

```
#hue::-webkit-slider-thumb {
  -webkit-appearance: none;
  width: 10px;
  height: 26px;
  border: 3px solid black;
  border-radius: 3px;
  cursor: ew-resize;
}

#formats {
  grid-area: formats;
}
```

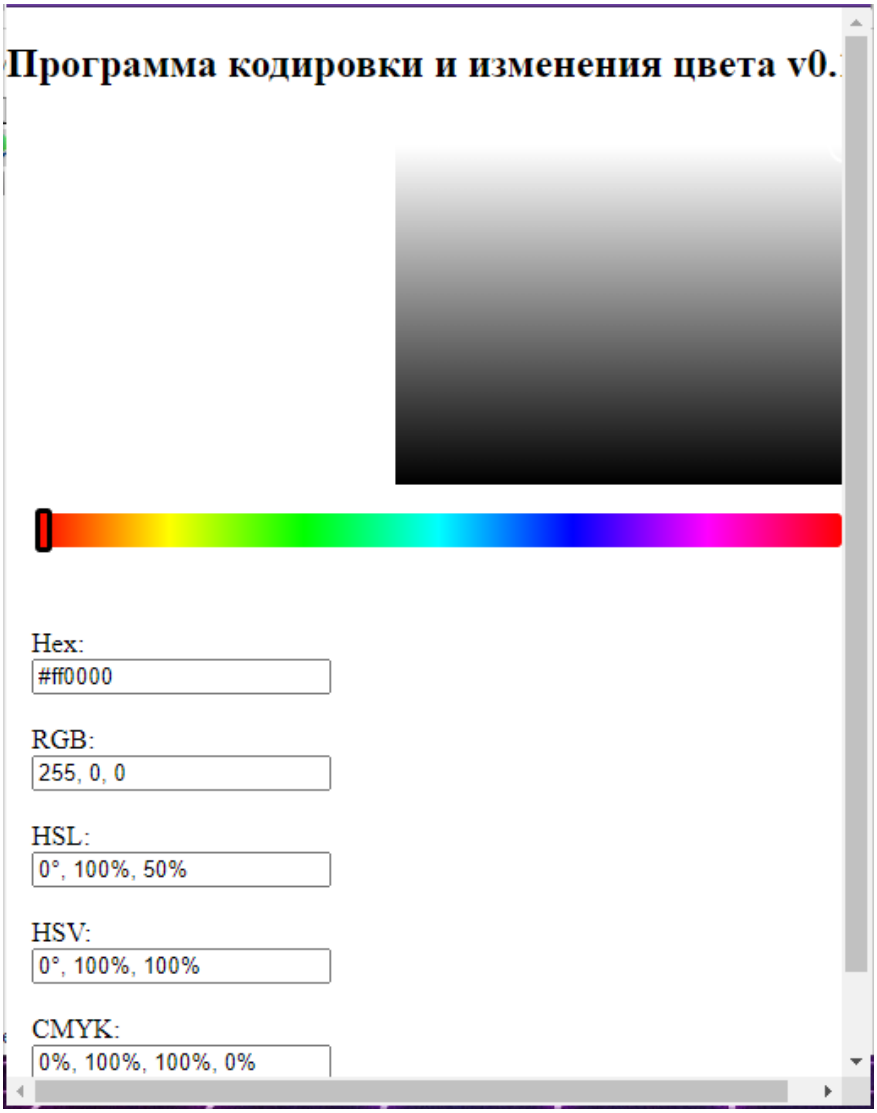


Рисунок 23 – Фрагмент веб-страницы

7 инкремент

В программный модуль добавлена функция вычисления координат цвета в модели HLS. Скриншот среды разработки (Рисунок 24) и код программного модуля представлены ниже.

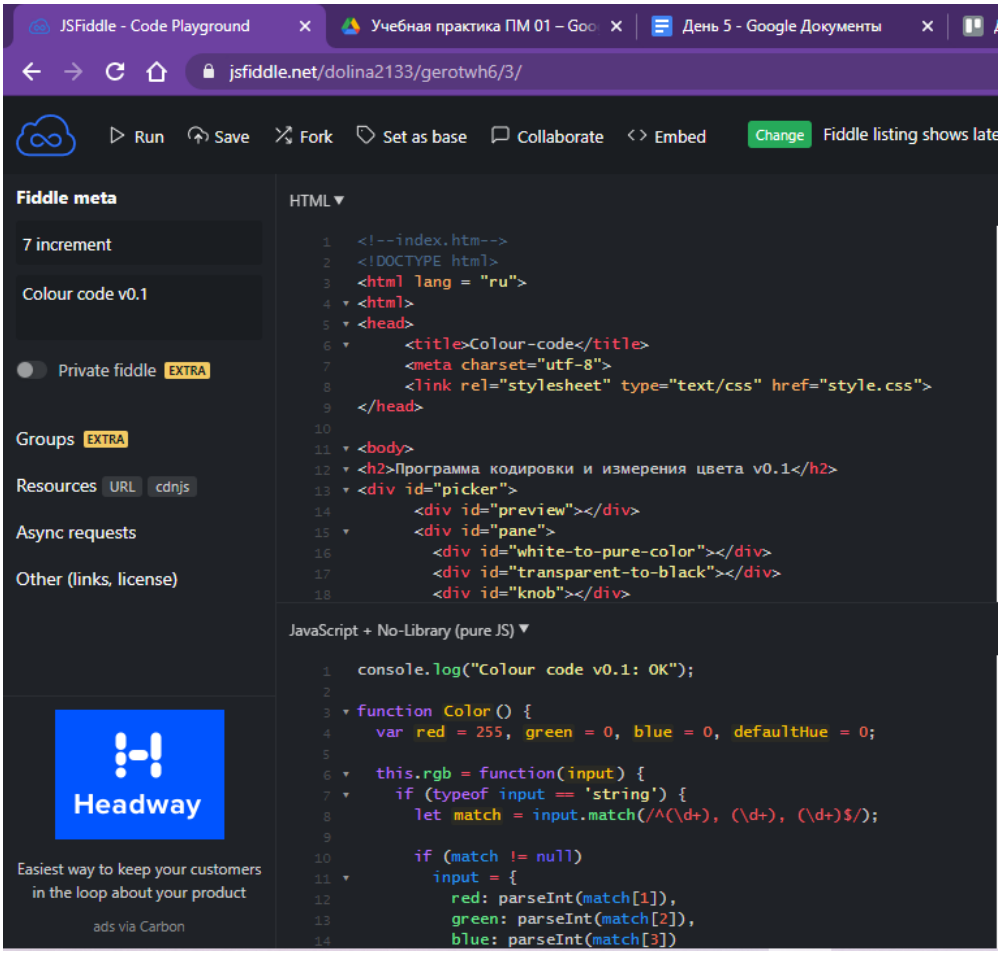


Рисунок 24 – 7 инкремент

//colour.js

```
// HSL color model (hue, saturation, lightness)
  this.hsl = function(input) {
    if (typeof input == 'string') {
      let match = input.match(/^(\d+)°, (\d+)%, (\d+)%$/);
      if (match != null)
        input = {
          hue: parseInt(match[1]),
          saturation: parseInt(match[2]),
          lightness: parseInt(match[3])
        };
    }
    if (typeof input == 'object') {
```

```
      var defaults = this.hsl(),
          valid = [['hue', 360], ['saturation', 100],
['lightness', 100]]
            .map(([prop, maxVal]) => {
              let validProp = typeof input[prop] == 'number' &&
input[prop] >= 0 && input[prop] <= maxVal;
              if (!validProp) input[prop] = defaults[prop];
              return validProp;
            })
            .reduce((a,b) => a || b);
      if (valid) {
        let s = input.saturation / 100,
            l = input.lightness / 100,
            h = input.hue / 60,
            c = (1 - Math.abs(2 * l - 1)) * s,
            x = c * (1 - Math.abs(h % 2 - 1)),
            a =
[[c,x,0],[c,x,0],[x,c,0],[0,c,x],[0,x,c],[x,0,c],[c,0,x]],
            m = l - c / 2,
            [red, green, blue] = a[Math.ceil(h)].map(v => 255 *
(v + m));
        this.rgb({red: red, green: green, blue: blue});
        defaultHue = input.hue;
        return this;
      }
    }
    var rgb = this.rgb(),
        r = rgb.red / 255,
        g = rgb.green / 255,
        b = rgb.blue / 255,
        v = Math.max(r, g, b),
        c = v - Math.min(r, g, b),
        lightness = v - c / 2;
    if (c == 0) var hue = defaultHue;
    else if (v == r) var hue = (g < b) ? 360 + 60 * (g - b) / c
: 60 * (g - b) / c;
    else if (v == g) var hue = 60 * (2 + (b - r) / c);
    else var hue = 60 * (4 + (r - g) / c);
    var saturation = (lightness == 0 || lightness == 1) ? 0 : c
/ (1 - Math.abs(2 * lightness - 1)) * 100;
    return {
      hue: hue,
      saturation: saturation,
      lightness: lightness * 100,
      toString: () => Math.round(hue) + '°, ' +
Math.round(saturation) + '%, ' + Math.round(lightness * 100) +
'%'
    };
}
}
```

8 инкремент

В программный модуль добавлена функция вычисления координат цвета в модели CMYK. Ниже представлен скриншот среды разработки (Рисунок 25) и код программного модуля.

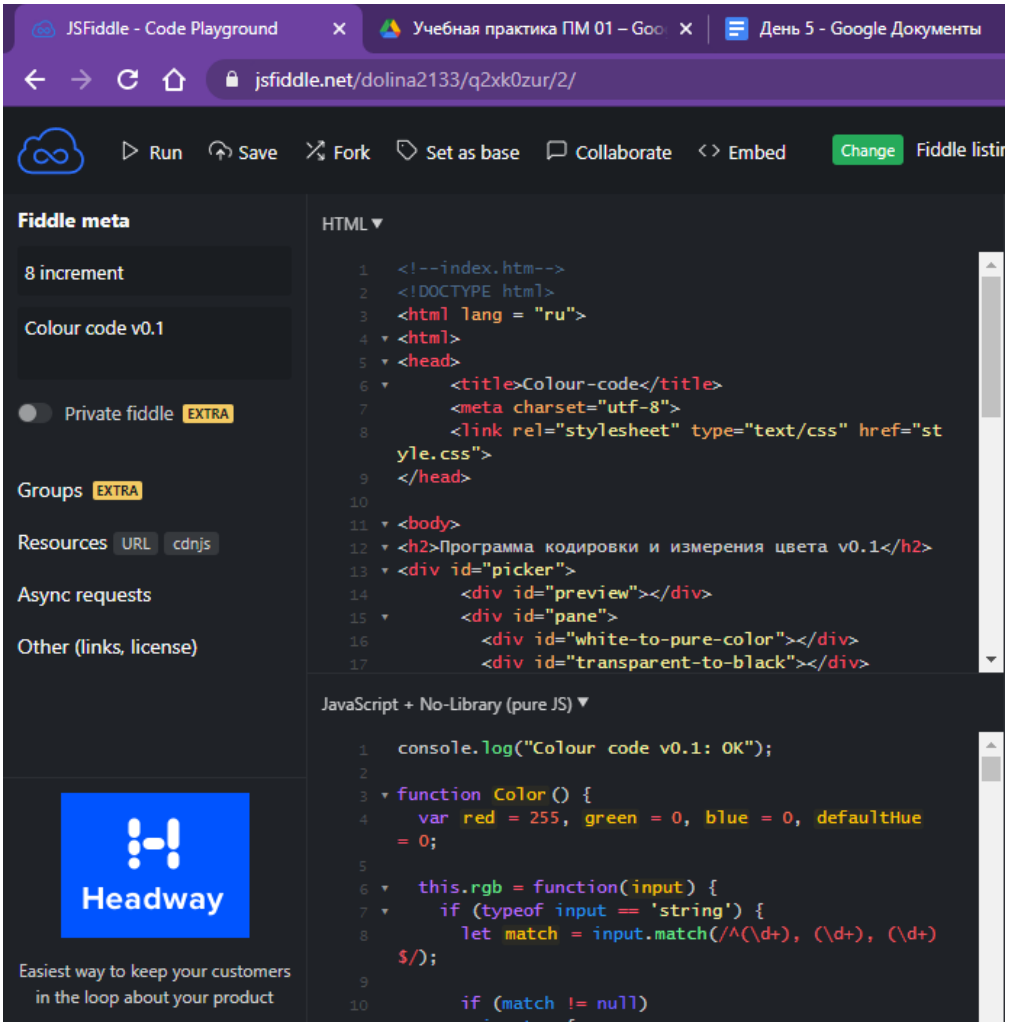


Рисунок 25 – 8 инкремент

//colour.js

```
// CMYK color model (cyan, saturation, yellow, key/black)
  this.cmyk = function(input) {
    if (typeof input == 'string') {
      let match = input.match(/^(\d+)%, (\d+)%, (\d+)%$/);

      if (match != null)
        input = {
          cyan: parseInt(match[1]),
          magenta: parseInt(match[2]),
          yellow: parseInt(match[3]),
```

```
          key: parseInt(match[4])
        };
    }

    if (typeof input == 'object') {
      let defaults = this.cmyk(),
          valid = ['cyan', 'magenta', 'yellow', 'key']
            .map((prop) => {
              let validProp = typeof input[prop] == 'number' &&
input[prop] >= 0 && input[prop] <= 100;
              input[prop] = (validProp ? input[prop] :
defaults[prop]) / 100;
              return validProp;
            })
            .reduce((a, b) => a || b);

      if (valid) {
        this.rgb({
          red: 255 * (1 - input.cyan) * (1 - input.key),
          green: 255 * (1 - input.magenta) * (1 - input.key),
          blue: 255 * (1 - input.yellow) * (1 - input.key)
        });
        defaultHue = this.hsv().hue;
        return this;
      }
    }

    var rgb = this.rgb(),
        r = rgb.red / 255,
        g = rgb.green / 255,
        b = rgb.blue / 255,
        key = 1 - Math.max(r, g, b),
        cyan = key == 1 ? 0 : (1 - r - key) / (1 - key) * 100,
        magenta = key == 1 ? 0 : (1 - g - key) / (1 - key) *
100,
        yellow = key == 1 ? 0 : (1 - b - key) / (1 - key) * 100;

    return {
      cyan: cyan,
      magenta: magenta,
      yellow: yellow,
      key: key * 100,
      toString: () => Math.round(cyan) + '%, ' +
Math.round(magenta) + '%, ' + Math.round(yellow) + '%, ' +
Math.round(key * 100) + '%'
    };
  }
}
```

9 инкремент

В программный модуль добавлена функция вычисления шестнадцатеричного кода цвета в формате RGB. Скриншот среды разработки (Рисунок 26) и код программного модуля представлены ниже.

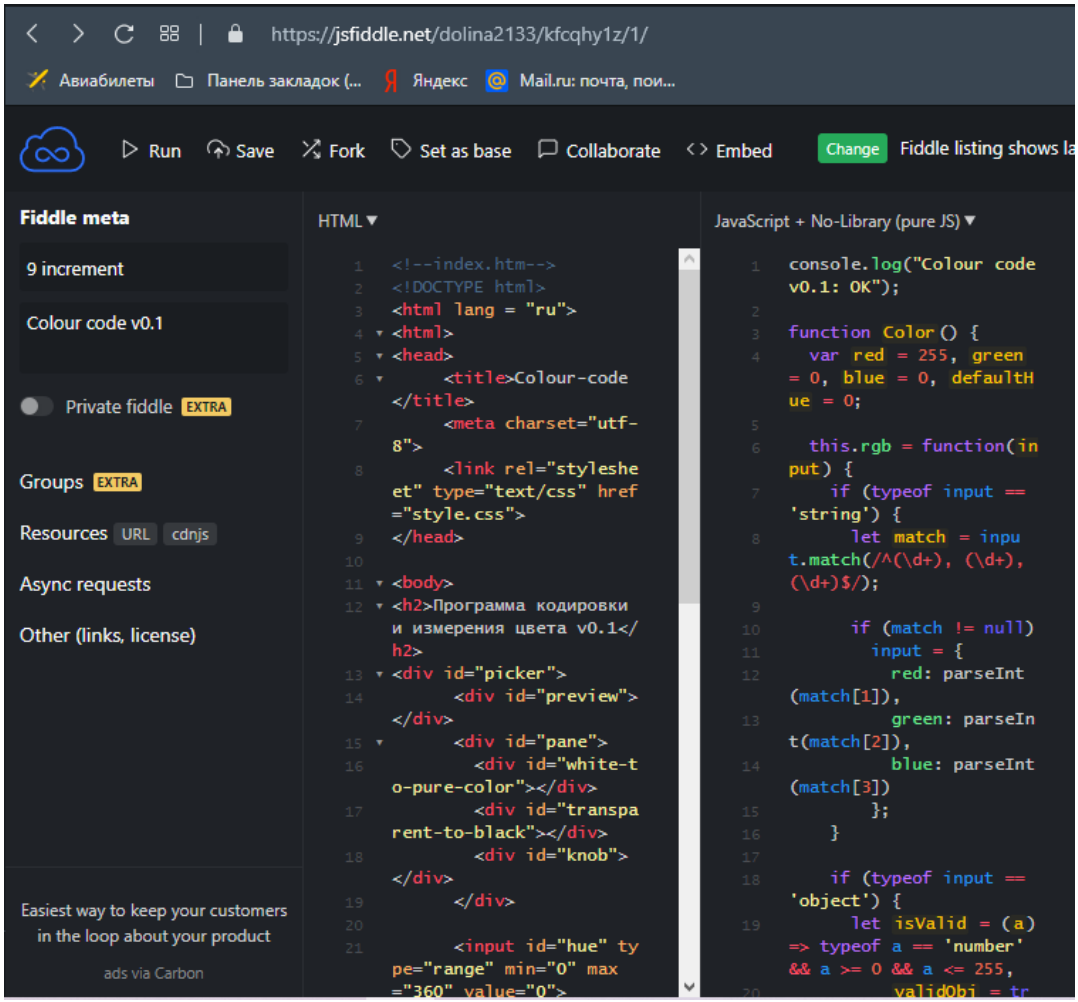


Рисунок 26 – 9 инкремент

//colour.js

```
// Hexadecimal color format
  this.hex = function(hex) {
    if (typeof hex == 'string' && hex.match(/^#[\dabcdef]{6}$/)
!= null) {
      var red = parseInt(hex.slice(1, 3), 16),
          blue = parseInt(hex.slice(3, 5), 16),
          green = parseInt(hex.slice(5, 7), 16);
      this.rgb({
        red: red,
        blue: blue,
```

```
        green: green
      });
      defaultHue = this.hsv().hue;
      return this;
    } else {
      var rgb = this.rgb(),
          redHex = Math.round(rgb.red).toString(16).padStart(2,
0),
          greenHex =
Math.round(rgb.green).toString(16).padStart(2, 0),
          blueHex =
Math.round(rgb.blue).toString(16).padStart(2, 0);
      return '#' + redHex + greenHex + blueHex;
    }
  }
};
```

10 инкремент

В структуру проекта добавлен файл библиотеки jQuery v.3.6.0, фрагмент которого показан на Рисунке 27.

Рисунок 27 – Файл библиотеки jQuery

11 инкремент

В структуру программного проекта добавлен программный модуль script.js, в котором с использованием библиотеки jQuery созданы селекторы для элементов графического интерфейса. С помощью события .on и .blur реализовано обновление форматов цвета. Скриншот среды разработки (Рисунок 28) и код программного модуля представлены ниже.

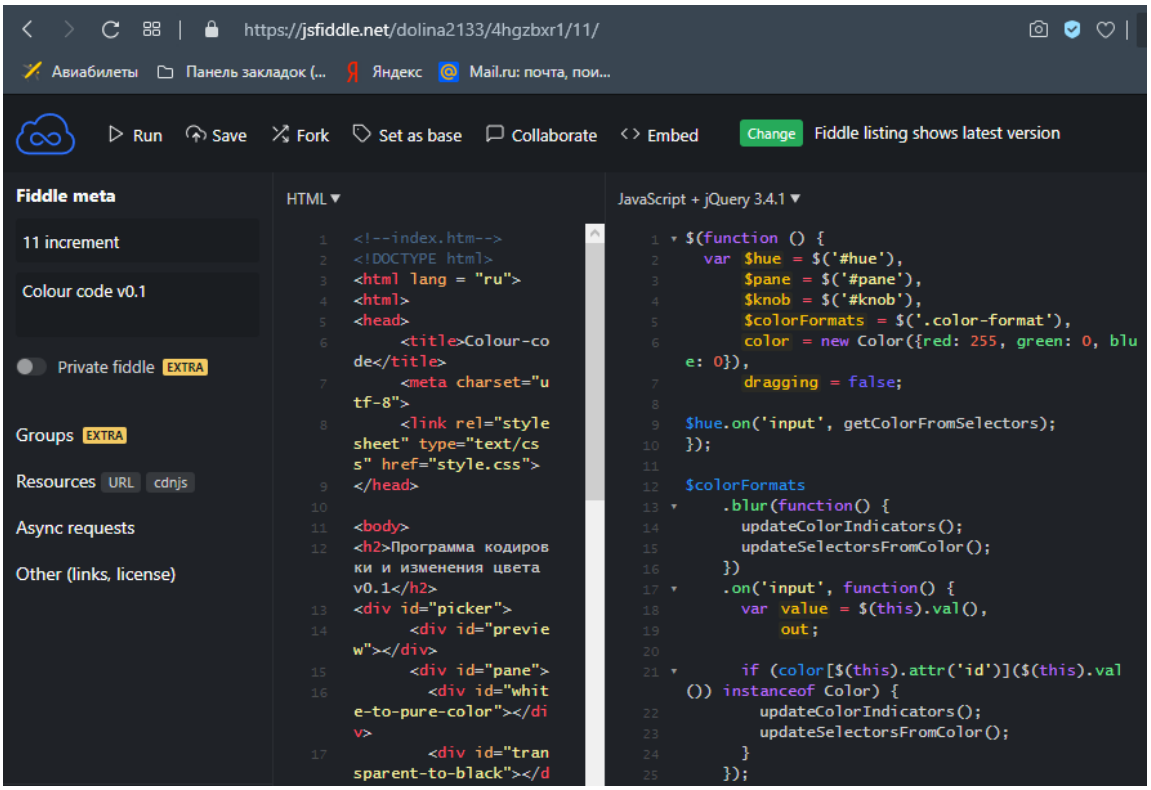


Рисунок 28 – 11 инкремент

//script.js

```
$(function () {
  var $hue = $('#hue'),
      $pane = $('#pane'),
      $knob = $('#knob'),
      $colorFormats = $('.color-format'),
      color = new Color({red: 255, green: 0, blue: 0}),
      dragging = false;
$hue.on('input', getColorFromSelectors);
});
$colorFormats
    .blur(function() {
      updateColorIndicators();
      updateSelectorsFromColor();
```

```
    })
    .on('input', function() {
      var value = $(this).val(),
          out;
      if (color[$(this).attr('id')]($(this).val()) instanceof
Color) {
        updateColorIndicators();
        updateSelectorsFromColor();
      }
    });
```

12 инкремент

С помощью функций библиотеки jQuery в программном модуле script.js реализовано обновление позиции движка слайдера и выбранного цвета. Скриншот среды разработки (Рисунок 29), веб-страница (Рисунок 30) и код программного модуля представлены ниже.

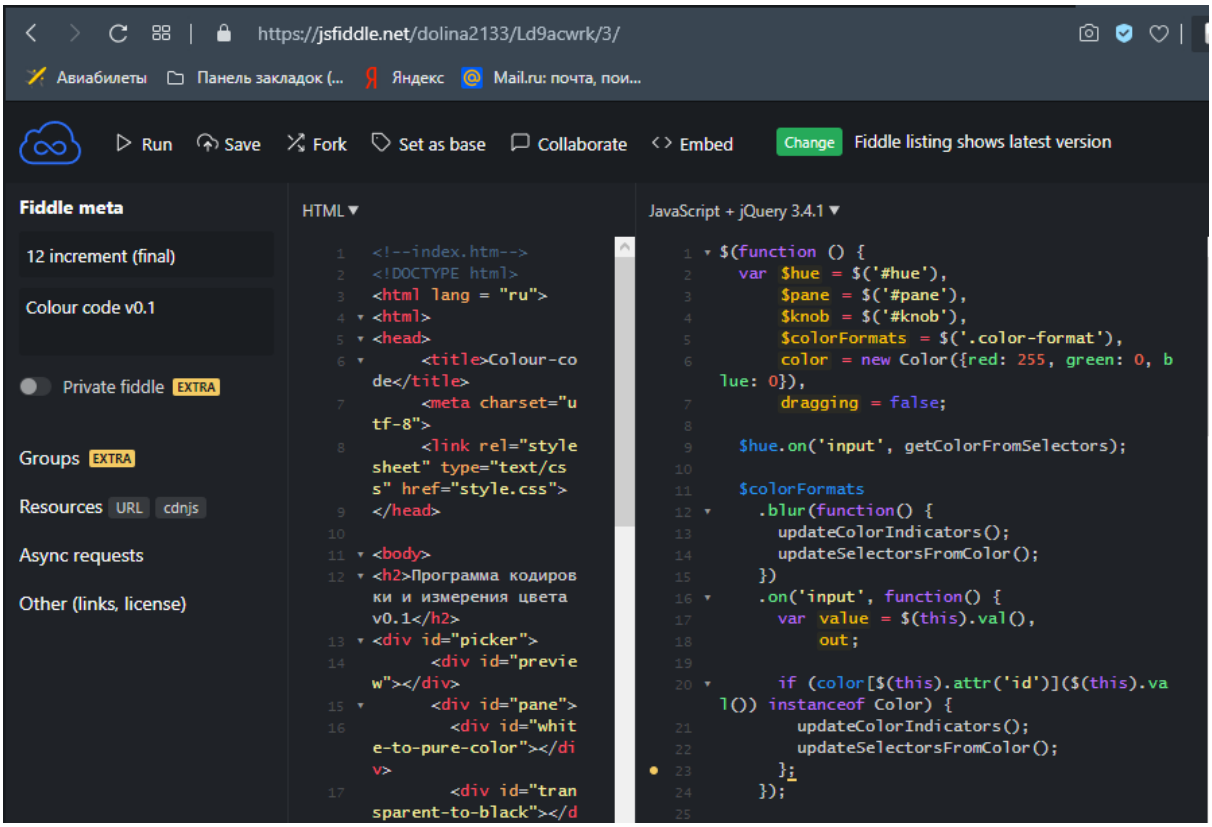


Рисунок 29 – 12 инкремент

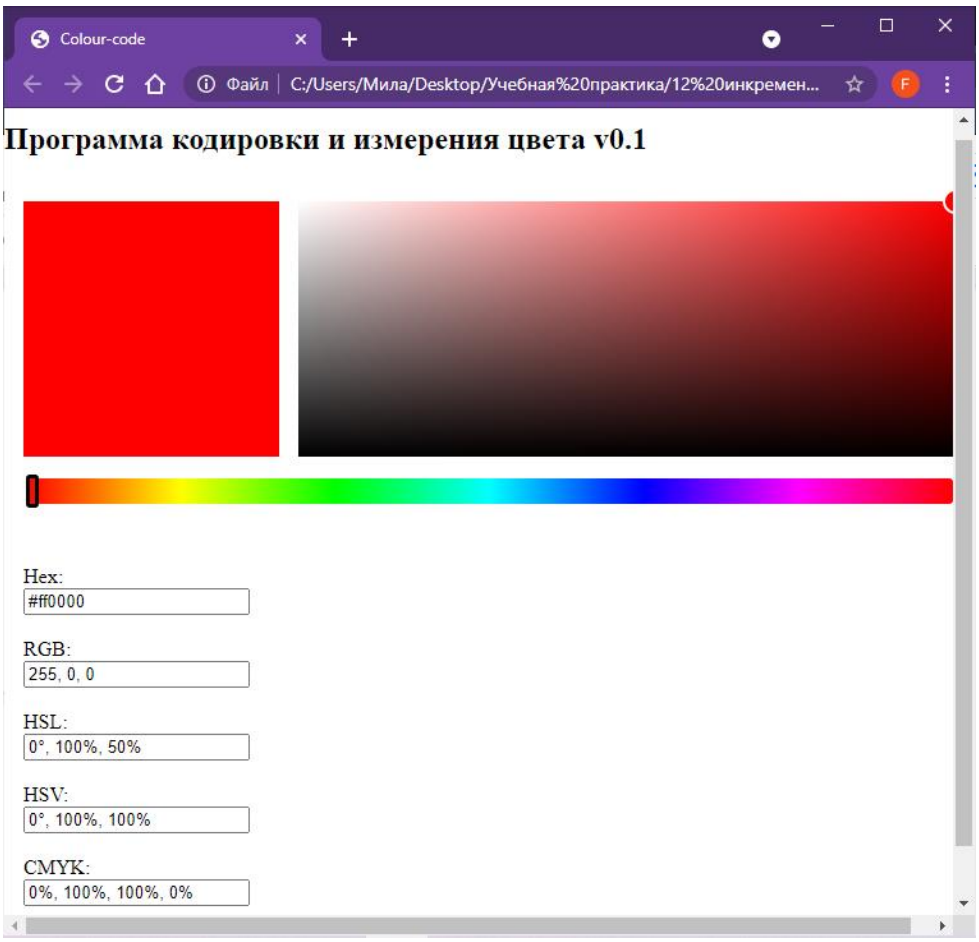


Рисунок 30 – Веб-страница