МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

СЕВЕРО-ВОСТОЧНЫЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ЦИФРОВЫХ ТЕХНОЛОГИЙ И ЭКОНОМИКИ

Кафедра математики и информатики

ОТЧЕТ
по учебной (вычислительной) практике

начало практики 30.06.2020, длительность практики – 2 недели

Тема «Разработка расширения для браузеров на движке Chromium для доступа в электронную почту»

Выполнил: студент 3 курса
гр. ПиБ-71
Исмайлов А. Э.

Проверил: доцент кафедры
математики и информатики
к.т.н. Сироткин А.В.

Магадан
2020

Оглавление

Введение

С каждым днем у человека появляется всё больше дел и работы. И многие вещи занимают большое количество времени и отвлекают от основной деятельности. Для оптимизации времени и увеличения производительности люди автоматизируют своё рабочее место. Пользователи, которые проводят большинство своего времени за компьютером, стараются держать все полезные инструменты под рукой. К таким инструментам относятся почта, которая позволяет обмениваться сообщениями и получать своевременно важную информацию. Но обычная почта через вкладку занимает много времени, так как не у всех хорошее соединение с интернетом и нету времени каждый раз обновлять содержимое страницы. Для этих целей были придуманы приложения почты, которые уведомляют звуком и показывают откуда пришло письмо. С таким приложением можно не отвлекаться на спам, сократить время обновления страницы и быть уверенным, что пользователь не пропустит важное сообщение. Целью данной работы является разработка расширения для браузера на движке Chromium «Проверка почты» с помощью средств HTML, CSS, JavaScript.

Результатом данной работы является расширение для Google Chrome, позволяющее пользователю экономить много времени на проверку почты, тем самым сосредоточить своё время на более важных вещах.

Для разработки данного расширения были выполнены следующие шаги:

1. Изучить литературу о JavaScript.
2. Формирования структуры расширения.
3. Изучение API выбранной почты.
4. Описание файла-манифеста manifest. json.
5. Создание итогового интерфейса
6. Разработка логики расширения и взаимодействия графических элементов друг с другом.
7. Отладка и тестирование разработанного расширения.

Для разработки данного расширения был выбран редактор кода «Web Storm», который имеет большой функционал с работой JavaScript – подсветка кода, проверкой на ошибки и быстрого выбора команд.

1. Создание структуры расширения

Для создания расширения были созданы следующие файлы:

1. manifest.json – данный файл создан для описания расширения и какие разрешения доступны ему;
2. popup.html и popup.css – данные файлы отвечают за внешний вид расширения. Первый файл отвечает за разметку и структуру расширения открываемого окна. Второй файл создан для определения стиля и графических элементов окна который видит пользователь.
3. Popup.js – этот скрипт отвечает за прослушивания всех элементов расширения с которыми взаимодействует пользователь. У данного скрипта есть модули который он вызывает:
    - punycode.min.js – библиотека для перевода с UNICODE в ASCII, данный скрипт предназначен для преобразования доменных имен в последовательность ASCII – символов.
    - lib.js – данный скрипт декодирует ASCII в UNICODE для отображения русского языка в письмах.
4. background.html – данная страница работает в фоновом режим. Она вызывает скрипты, выполняет соединения с почтой и обрабатывает запросы пользователя. У данной страницы есть модули, которые он вызывает:
    - sound.js – данный скрипт хранит в локальном хранилище оповещающий звук и воспроизводит его при поступления сообщения;
    - utils.js - данный скрипт обращается к API почты и возвращает данные полученные с API.;
    - view.js – данный скрипт показывает количество непрочитанных сообщений, вызывает уведомления и проигрывает звук при получении сообщения;
    - sentMSG – данный скрипт отвечает за переход по ссылкам при нажатии определённой кнопки;
    - account.js – данный скрипт работает с API Mail.ru, он возвращает данные об аккаунте и проверят каждый 5 секунд авторизацию.
    - user.js – данный скрипт работает с API для получение токена, статуса, раздела сообщений, сообщения, непрочитанные сообщения. При выходе из почты они обнуляются.
    - main.js – скрипт который подключает все модули воедино и использует локальное хранилище.

2. Создание файла manifest.json

Для каждого расширения на движке Chronium нужен специальный файл manifest.json.

Данный файл описывает расширение, определяет его разрешения и его структуру. Без данного файла браузер не запустит расширение и не примет его.(листинг 1).

Листинг 1 – Содержимое файла manifest.json

```
{
   "background": {
      "page": "background.html"
   },
   "browser_action": {
      "default_icon": "img/ico_panel.png",
      "default_title": "Проверить почту",
      "default_popup": "popup.html"
   },
   "icons" : {
      "16" : "img/ico_panel.png",
      "48" : "img/ico_panel.png",
      "128" : "img/ico_panel.png"
   },
   "default_locale": "ru",
   "description": "Расширение для проверки почты",
   "manifest_version": 2,
   "version": "1.0",
   "name": "Проверить почту",
   "permissions": [ "idle", "notifications", "https://mail.ru/*",
}
```

3. Создание итогового интерфейса

Разметка страницы состоит из следующих блоков:

- Head - в данном блоке располагается название расширения, подключения CSS – стилей и скриптов.
- В контейнере div названный header хранятся ссылки на главную страницу Mail.ru и Обновления списка. Так же находится контейнер с контентом (авторизация и письма).

Для разработки дизайна интерфейса использовались средства CSS, позволяющие работать с документами, описанными с помощью языка HTML. (рис. 1).

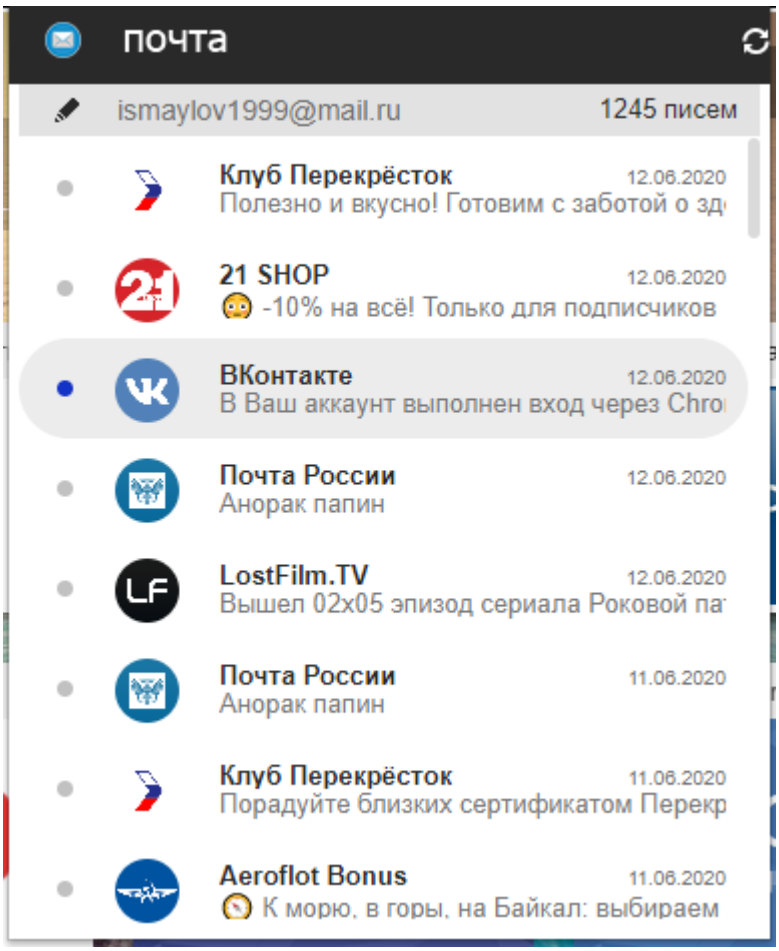

Рис. 1. Интерфейс расширения

4. Разработка логики расширения

Основным скриптом работы расширения является Users.js получения данных при обращении к API. (Листинг. 2).

Листинг 2 – пример получения токена с помощью GET запроса к API.

```
User.prototype.getToken = function() {
    var url = 'https://mailru-checker-api.e.mail.ru/api/v1/tokens?email=' + this.email + '&x-email=' + this.email;
    app.utils.request( options: {
        url: url,
        method: "GET"
    }, this.parsTokenResult.bind(this), this.errorRequestToken.bind(this))
};

User.prototype.getTokenSdc = function() {
    var url = 'https://mailru-checker-api.e.mail.ru/api/v1/tokens?email=' + this.email + '&x-email=' + this.email;
    var urlSdc = 'https://auth.mail.ru/sdc?from=' + decodeURIComponent(url);
    app.utils.request( options: {
        url: urlSdc,
        method: "GET"
    }, this.parsTokenSdcResult.bind(this), this.errorRequestToken.bind(this))
};
```

Таким же образом получаем и другие данные о пользователе, его профиле и почте.

В скрипте Utils.js хранится шаблон запроса к API. (Листинг 3).

Листинг 3 – шаблон запроса.

```
Utils.prototype.request = function( options, callback, errorCallback ) {
    let xhr = new XMLHttpRequest(),
        url = options.url,
        params = options.params || null,
        method = options.method || 'GET',
        header = options.header || null;

    xhr.open(method, url, async: true);
    xhr.onreadystatechange = function() {
        if (this.readyState == 4) {
            if (this.status == 200) {
                if(callback) {
                    let respons = this.response || this.responseText;
                    callback(respons);
                }
            } else {
                if(errorCallback) {
                    errorCallback();
                }
            }
        }
    };

    if (header) {
        for (var key in header) {
            xhr.setRequestHeader(key, header[key]);
        }
    }

    xhr.send(params);
};
```

Функция, отвечающая за вывод непрочитанных сообщений если пользователь не авторизирован, если пользователь не авторизовался – выводит сообщение о том, что он не зашел в почту. (Листинг 4).

Листинг 4 – Вывод сообщений.

```
function viewMesg() {
    var users = backgroundPage.app.account.users;
    var flagAuth = false;
    document.getElementById( elementId: 'content').innerHTML = '';

    for(var key in users) {
        var userItem = users[key];

        if(users[key].status) {
            flagAuth = true;
            insertMess(userItem.messages, userItem.email, userItem.guid, userItem.count);
            addEvent(userItem.email, userItem.guid);
        }
    }

    if(!flagAuth) {
        notAuth();
    }
}
```

5. Тестирование и откладка расширения

Для проверки работоспособности расширения, было отослано пару новых писем.

До и после отправки тестовых сообщений (рис. 2.).

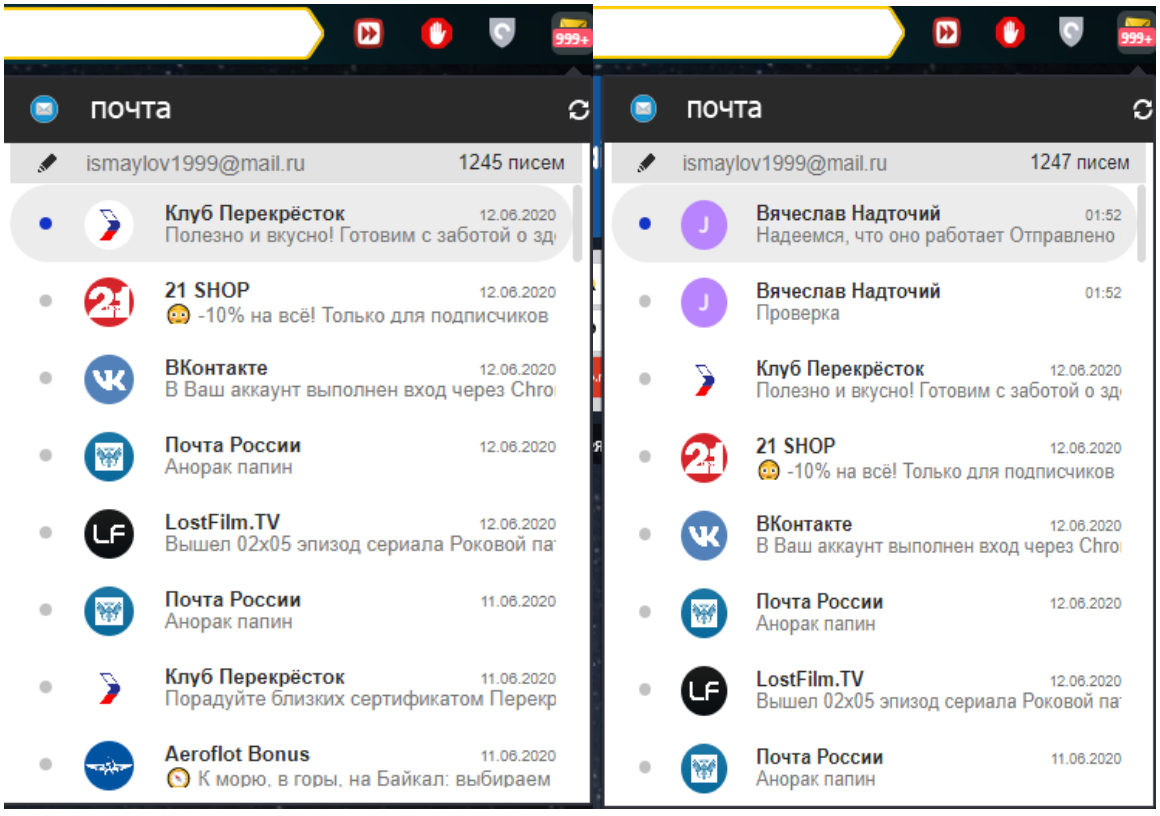

Рис. 2. Результат отправки сообщений

Так же пришло звуковое и текстовое уведомление о новом сообщении с выбором варианта ответа (рис. 3)

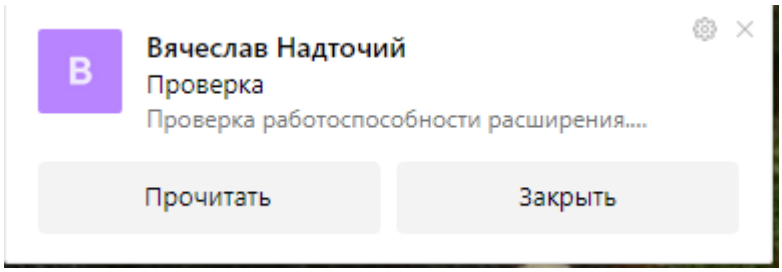

Рис. 3. Уведомление

Заключение

Было разработано расширение, которое проверяет почту, в ходе его написания было изучено основы работы с расширениями браузера и взаимодействия с API (application programming interface).

Расширение «Проверка почты» предоставляет пользователю быстрый доступ к почте в любое время в любой вкладке браузера, а также позволяет получать уведомления о новых письмах.

Все задачи, поставленные в начале практической работы, были выполнены, следовательно, ее цель – разработка расширения для браузеров на движке Chromium «Проверка почты» с помощью средств HTML, CSS, JavaScript, была выполнена.

Список источников

1. Современный учебник JavaScript: [Электронный ресурс] URL: https://learn.javascript.ru/ (дата обращения 01.07.2020).
2. Создание расширений для Google Chrome : [Электронный ресурс] URL : https://legacy.gitbook.com/book/torus/sozdanie-rasshirenii-dlya-google-chrome/details (дата обращения 01.07.2020).
3. Руководство по использованию JS AP Mail.ruI: [Электронный ресурс] URL : https://api.mail.ru/docs/guides/jsapi/ (дата обращения 03.07.2020).
4. Справочник по JS API Mail.ru: [Электронный ресурс] URL : https://api.mail.ru/docs/reference/js/ (дата обращения 03.07.2020).
5. Авторизация для сайтов Mail.ru: [Электронный ресурс] URL : https://api.mail.ru/docs/guides/oauth/sites/ (дата обращения 04.07.2020).