

Создание приложения нахождения хеша на Electron.js

Кизьянов Антон Олегович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье будет рассказано что такое Electron.js где его взять и создано приложение для нахождения хеш сумм от введенного текста.

Ключевые слова: Electron.js, JavaScript, HTML5

Creating a hash finding application on Electron.js

Kizyanov Anton Olegovich

Sholom-Aleichem Priamursky State University

student

Abstract

This article will tell you what Electron.js is, where to get it and create an application to find the hash of the sums from the entered text.

Keywords: Electron.js, JavaScript, HTML5

Web технологии уже давно вышли за рамки всем привычного web'а. После создания JavaScript движка Node.js стало доступно выполнять JavaScript не только в браузерах, а и на самой операционной системе. Это стало бумом развития всяческих библиотек и фреймворков работающих на данном движке, одним из которых и является Electron.js.

Electron.js позволяет разрабатывать настольные графические приложения с использованием веб-технологий: он сочетает в себе механизм рендеринга Chromium и среду выполнения Node.js.[1]

Цель исследования – написание приложения для нахождения разных сумм хешей с использованием библиотеки Electron.js.

Ранее этим вопросом интересовались А.А. Крюкова, В.А. Почебут развивали тему «Особенности тестирования, внедрения и разработки десктопных приложений» [2] в которой при анализе последних исследований, была поставлена проблема о необходимости внедрения десктопных приложений. В связи с этим, предложены пути решения данной проблемы, на основе детального изучения алгоритма внедрения и анализе платформ: nw.js, electron.js, cuba. Выделены преимущества данных приложений над различными web-приложениями. А.А. Гарифуллин с темой «Моделирование процессов добычи полезных ископаемых на веб-платформе» [3], а подробнее про основные вопросы, связанные с темой моделирования в веб-среде: платформа Electron.js, компоненты

математической модели, а также применение микросервисной архитектуры в веб. М.В. Маревская и И.Б. Государев опубликовали статью «Особенности использования веб-фреймворков при разработке десктопных приложений» [4] рассказали про фреймворки, позволяющие создавать десктопные приложения на основе веб-технологий, рассмотрены фреймворки Electron и NW.js, их общие черты, специфика их использования, различия между ними.

Electron.js позволяет создавать настольные приложения на чистом JavaScript, предоставляя среду выполнения со встроенными в систему API-интерфейсами.

Перед началом работы нужно установить Node.js[5] среду выполнения JavaScript кода. Она устанавливается вместе с NPM(пакетный менеджер), он позволяет устанавливать библиотеки одной командой.

Для начала нужно ввести команду **npm init** в директории будущего проекта. Она создаст файл `package.json` в котором будут находиться зависимости.

Следующей командой будет **npm install electron**, npm установит `electron.js` в конкретную директорию.

Теперь нужно отредактировать файл `package.json`, в него нужно добавить поля `main` и `scripts`, они определяют главный файл и сразу прописывают команду на запуск приложения. Должно выглядеть так-же как показано ниже.

```
{
  "name": "hash",
  "version": "0.0.1",
  "description": "",
  "main": "app.js",
  "scripts": {
    "start": "electron app.js"
  },
  "devDependencies": {
    "electron": "^3.0.2"
  }
}
```

Далее необходимо создать файл `app.js` рядом с файлом `package.js` и поместить в него следующее содержимое.

```
const {app, BrowserWindow} = require('electron');
const path = require('path');
const url = require('url');

let window = null;

app.once('ready', () => {
  window = new BrowserWindow({
    width: 800,
    height: 600,
    backgroundColor: "#D6D8DC",
    show: false
  });
});
```

```

    });
    window.loadURL(url.format({
      pathname: path.join(__dirname, 'index.html'),
      protocol: 'file:',
      slashes: true
    }));

    window.once('ready-to-show', () => {
      window.show()
    })
  });

```

Этот код формирует главное окно приложения, описывает его размеры и цвет, а потом загружает файл index.html.

Файл index.html это главная страница приложения, уже к ней подключаются все библиотеки и стили. Необходимо создать его и поместить следующее содержимое в него.

```

<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="utf-8">
  <title>Hash</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script>
    delete module.exports
  </script>
  <script src="https://code.jquery.com/jquery-3.2.1.js"></script>
  <script src="./window.js" charset="utf-8"></script>
</head>
<body>
<div id="container" class="container-fluid">
  <div class="row">
    <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12 ">
      <h3 class="hash-heading">Input</h3>
      <textarea rows="2" id="text-input" class="form-control text-input"
placeholder="Введите текст и посмотрите на его
xew"></textarea>
    </div>
  </div>
  <div class="row">
    <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12">
      <h3 class="hash-heading">MD5</h3>
      <pre id="md5-output" class="hash-output"> </pre>
    </div>
  </div>
  <div class="row">
    <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12">
      <h3 class="hash-heading">SHA-1</h3>
      <pre id="sha1-output" class="hash-output"> </pre>
    </div>
  </div>
  <div class="row">
    <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12">
      <h3 class="hash-heading">SHA-256</h3>
      <pre id="sha256-output" class="hash-output"> </pre>
    </div>
  </div>
</div>

```

```
<div class="row">
  <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12">
    <h3 class="hash-heading">SHA-512</h3>
    <pre id="sha512-output" class="hash-output"> </pre>
  </div>
</div>
</div>
</body>
</html>
```

В файле index.html подключаются CSS библиотека bootstrap для красивого вида страницы, загружается библиотека JQuery для быстрого манипулирования содержимым страницы и внешний модуль window.js в котором находится код нахождения хеша.

Далее требуется создать файл window.js и поместить следующее содержимое в него.

```
$(() => {
  const crypto = require('crypto');

  $('#text-input').bind('input propertychange', function () {
    const text = this.value;

    const md5 = crypto.createHash('md5').update(text,
'utf8').digest('hex');
    $('#md5-output').text(md5);

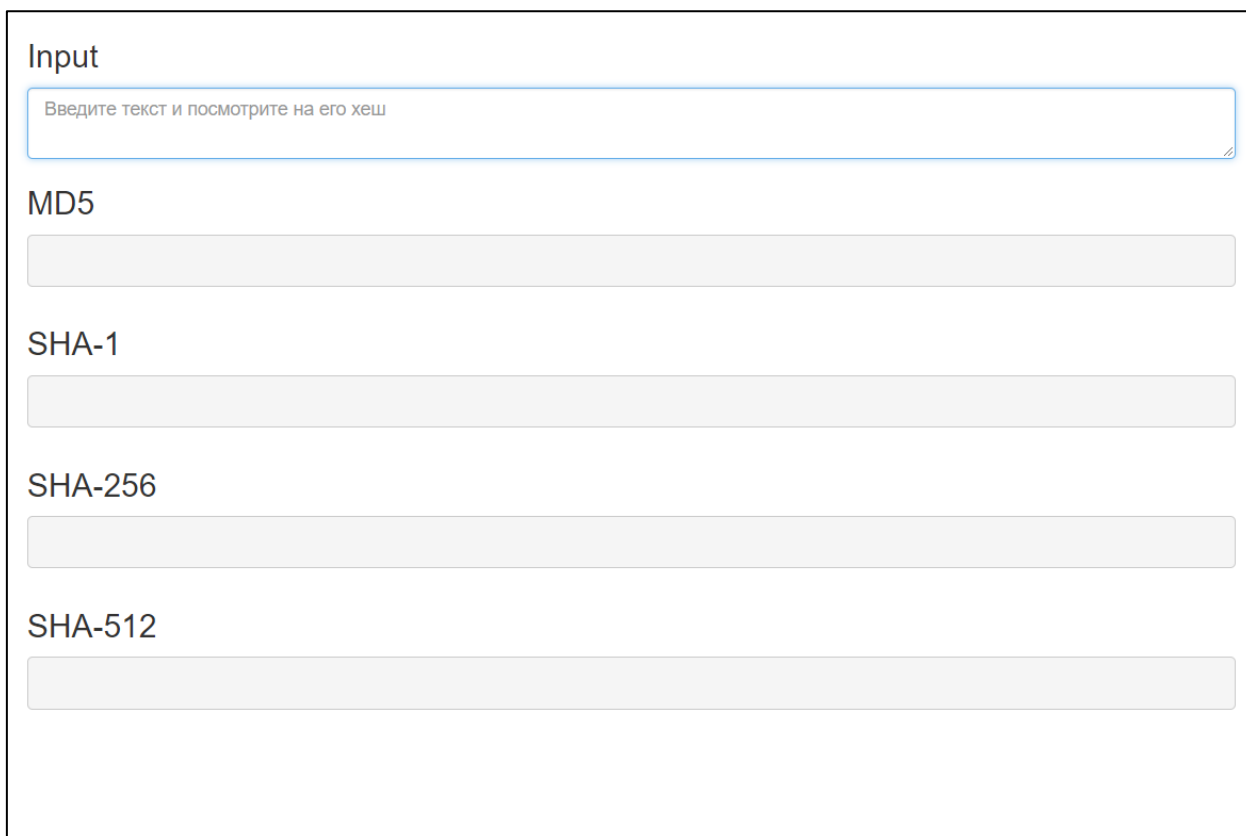
    const sha1 = crypto.createHash('sha1').update(text,
'utf8').digest('hex');
    $('#sha1-output').text(sha1);

    const sha256 = crypto.createHash('sha256').update(text,
'utf8').digest('hex');
    $('#sha256-output').text(sha256);

    const sha512 = crypto.createHash('sha512').update(text,
'utf8').digest('hex');
    $('#sha512-output').text(sha512)
  });

  $('#text-input').focus()
})
```

Этот код позволяет при каждом обновлении поля text-input обновлять значение каждого хеша по отдельности, а потом опять устанавливать фокус в поле. На данном этапе приложение готово, чтобы запустить нужно, открыть в консоли, запустить команду npm start находясь в этой директории. Результат представлен на рисунках 1 и 2.



Input

Введите текст и посмотрите на его хеш

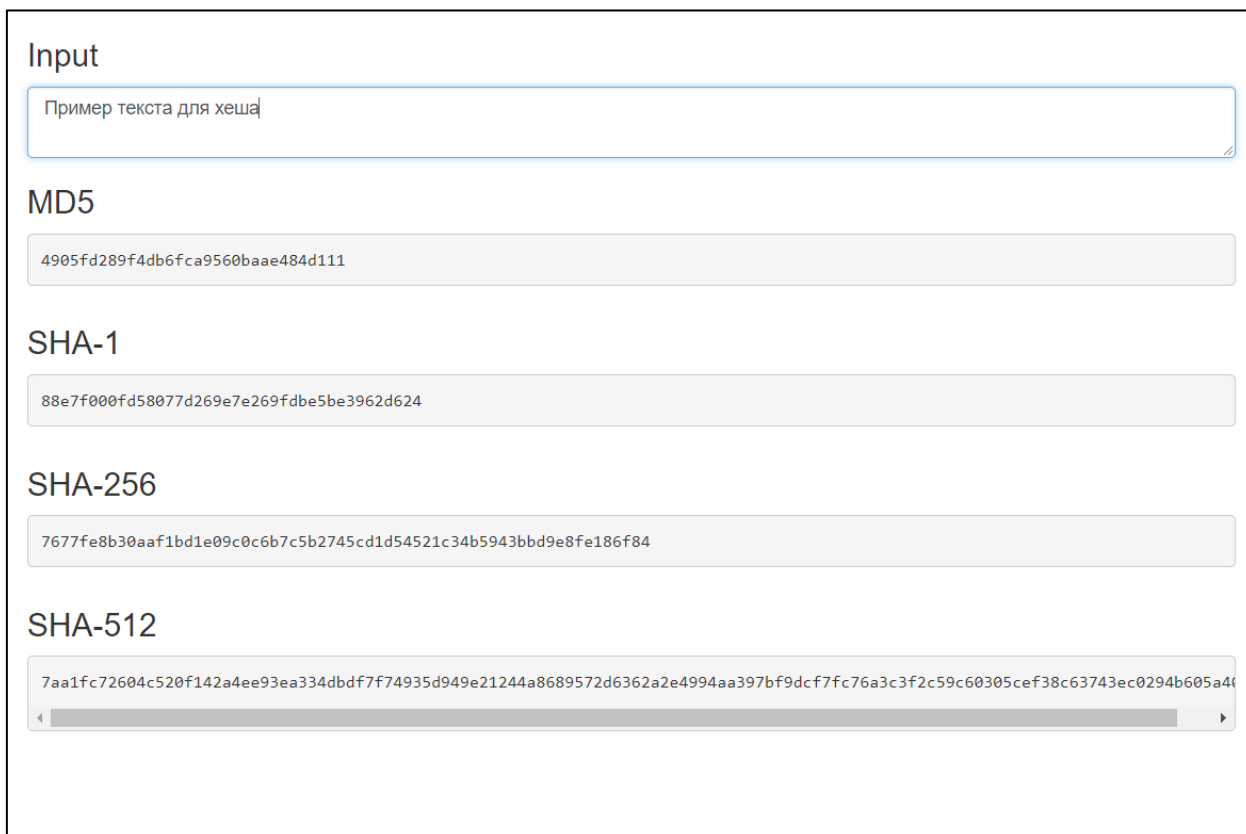
MD5

SHA-1

SHA-256

SHA-512

Рис. 1 Запущенное приложение



Input

Пример текста для хеша

MD5

4905fd289f4db6fca9560baae484d111

SHA-1

88e7f000fd58077d269e7e269fdba5be3962d624

SHA-256

7677fe8b30aaf1bd1e09c0c6b7c5b2745cd1d54521c34b5943bbd9e8fe186f84

SHA-512

7aa1fc72604c520f142a4ee93ea334dbdf7f74935d949e21244a8689572d6362a2e4994aa397bf9dcf7fc76a3c3f2c59c60305cef38c63743ec0294b605a44

Рис. 2 Вывод значения разных хешей

Вывод

Таким образом, сфера применения JavaScript растет и тем самым позволяет разрабатывать продукты на одном языке под платформы Windows и Web.

Библиографический список

1. Electron.js URL: <https://electronjs.org/> (Дата обращения: 07.08.2019)
2. Крюкова А.А., Почебут В.А. Особенности тестирования, внедрения и разработки десктопных приложений // Синергия Наук. 2017. № 12. С. 964-971. URL: <https://elibrary.ru/item.asp?id=29392793> (Дата обращения: 07.08.2019)
3. Гарифуллин А.А. Моделирование процессов добычи полезных ископаемых на веб-платформе // В сборнике: Альманах научных работ молодых ученых университета ИТМО XLVII научная и учебно-методическая конференция Университета ИТМО по тематикам: экономика; менеджмент, инноватика. 2018. С. 101-103. URL: <https://elibrary.ru/item.asp?id=36986963> (Дата обращения: 07.08.2019)
4. Маревская М.В., Государев И.Б. Особенности использования веб-фреймворков при разработке десктопных приложений В сборнике: Альманах научных работ молодых ученых университета итмо xlvii научная и учебно-методическая конференция Университета ИТМО по тематикам: экономика; менеджмент, инноватика. 2018. С. 208-210. URL: <https://elibrary.ru/item.asp?id=36987029> (Дата обращения: 07.08.2019)
5. Node.js. URL: <https://nodejs.org/en/> (Дата обращения: 07.08.2019)