SONIX CO., LTD.

SONIX R&D REPORT

2015 年 6 月 30 日

HTML5, ELECTRON

株式会社ソニックス　本社 R&D グループ

レポート概要	
R&D テーマ・目的	様々な業務システムにおける最新の Web 技術（HTML5, Node.js）の活用機会検討における技術調査。 本レポートでは、HTML5、Node.js を活用したクロスプラットフォームのデスクトップアプリケーション開発フレームワーク「Electron」を試用した結果を報告。
技術キーワード	HTML5、Node.js、デスクトップアプリケーション、クロスプラットフォーム、Electron
本レポート対象者	開発担当者、HTML5 導入検討に携わる企画担当者

Electron でクロスプラットフォームのデスクトップアプリを作る

Electron(旧 Atom Shell)は、Web 技術を使って、クロスプラットフォームのデスクトップアプリを作るためのフレームワークです。

元々は Atom editor を開発するために作られたフレームワークで、最近では Slack のクライアントアプリや、Visual Studio Code も Electron で実装されています。

この記事では Electron を使った、Todo アプリの作り方を説明します。

Electron の特徴

Electron には以下のような特徴があります。

- Web 技術
- HTML, CSS, Javascript(node.js)を使ってアプリを開発することができる。
- node.js の豊富なライブラリを活用できる。

- オープンソース
- Electron はオープンソースとして公開されていて、GitHub がメンテナンスしている。

- クロスプラットフォーム
- Mac, Windows, Linux 向けにビルドできる。

Electron のインストール

npm で electron をインストールします。

```
# Install the `electron` command globally
$ npm install electron-prebuilt -g
...

$ electron -v
v0.26.1
```

Todo アプリ作成

今回は簡単に作成するため、Backbone.js のサンプルプロジェクトの Todo アプリのソースコードを流用します。

このアプリの全てのソースコードは Github で公開しています。

Todo アプリのプロジェクトは以下の構造になっています。

```
todos
├── package.json
├── main.js
├── index.html
├── lib
│   └── todos.js
├── images
│   └── destroy.png
└── stylesheets
    └── todos.css
```

package.json のフォーマットは node.js のモジュールと同じです。main フィールドにアプリの起動スクリプトを指定します。 dependencies フィールドには依存するライブラリを記述します。

```
# package.json

{
  "name": "todos",
  "version": "0.1.0",
  "main": "main.js",
  "dependencies": {
    "jquery":                "2.1.4",
    "underscore":            "1.8.3",
    "backbone":              "1.2.0",
    "backbone.localstorage": "1.1.16"
  }
}
```

main.js ではウィンドウを作成し、システムイベントを処理します。今回は Quick Start のソースコードをそのまま使いました。

```
# main.js

var app = require('app');  // Module to control application life.
var BrowserWindow = require('browser-window');  // Module to create native br
owser window.
```

```
// Report crashes to our server.
require('crash-reporter').start();

// Keep a global reference of the window object, if you don't, the window wil
l
// be closed automatically when the javascript object is GCed.
var mainWindow = null;

// Quit when all windows are closed.
app.on('window-all-closed', function() {
  if (process.platform != 'darwin')
    app.quit();
});

// This method will be called when Electron has done everything
// initialization and ready for creating browser windows.
app.on('ready', function() {
  // Create the browser window.
  mainWindow = new BrowserWindow({width: 570, height: 600});

  // and load the index.html of the app.
  mainWindow.loadUrl('file://' + __dirname + '/index.html');

  // Open the devtools.
  mainWindow.openDevTools();

  // Emitted when the window is closed.
  mainWindow.on('closed', function() {
    // Dereference the window object, usually you would store windows
    // in an array if your app supports multi windows, this is the time
    // when you should delete the corresponding element.
    mainWindow = null;
  });
});
```

index.html は表示する Web ページです。script タグで lib/todos.js を読み込むようにしています。

```
# index.html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>Todos</title>
  <link rel="stylesheet" href="stylesheets/todos.css"/>
</head>

<body>
...
  <script src="lib/todos.js"></script>
...
  </body>
</html>
```

lib/todos.js には Todo アプリのメインロジックを記述します。backbone.js Todos のソースコードとほとんど同じですが、require で Node モジュールを読み込んでいるところが異なります。

```
# lib/todos.js

var $ = require('jquery')
var _ = require('underscore')
var Backbone = require('backbone')
Backbone.LocalStorage = require('backbone.localstorage')

// Load the application once the DOM is ready, using `jQuery.ready`:
$(function(){

  // Todo Model
  // ----------

  // Our basic **Todo** model has `title`, `order`, and `done` attributes.
  var Todo = Backbone.Model.extend({

...
```

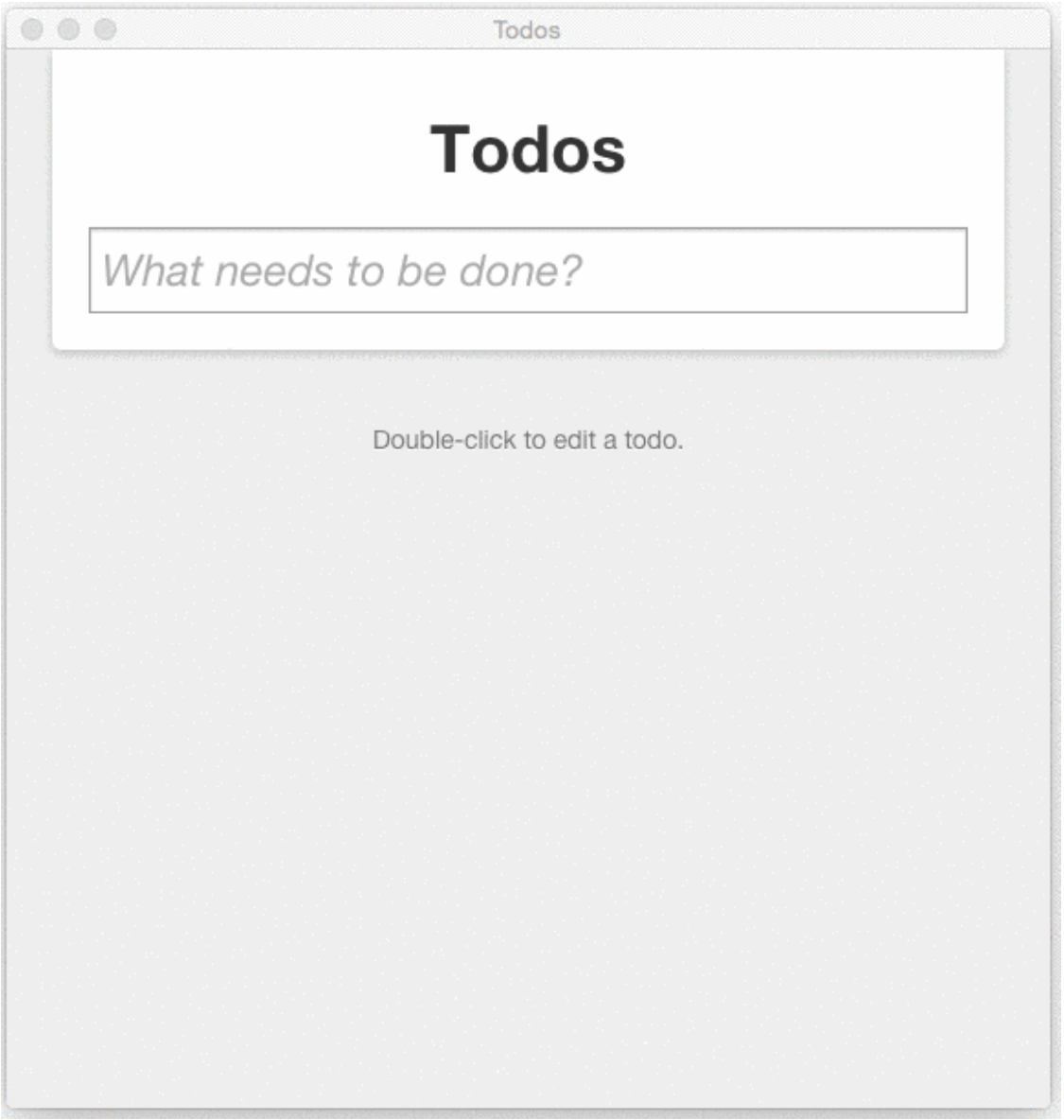
アプリの実行

最初に package.json に記述した依存ライブラリをインストールします。

```
$ cd todos
$ npm install
```

electron コマンドでアプリを実行します。引数にはプロジェクトのディレクトリを指定します。

```
$ electron .
```



上記の画面が表示されれば成功です。

アプリの配布

Mac 向けに配布する

1 . Electron のリリースページから electron-v0.26.1-darwin-x64.zip をダウンロードします。 2 . todo アプリケーションのディレクトリを Electron.app/Contents/Resources/app にコピーします。

```
$ cd todos
$ cp -r . ~/Downloads/electron-v0.26.1-darwin-x64/Electron.app/Contents/Resources/app
```

3 . アプリケーション名を Todos.app にリネームします

```
$ mv ~/Downloads/electron-v0.26.1-darwin-x64/Electron.app ~/Downloads/ele
ctron-v0.26.1-darwin-x64/Todos.app
```

4 . 作成した Todos.app を配布します。

Windows 向けに配布する

1 . Electron のリリースページから electron-v0.26.1-win32-x64.zip をダウンロードします。(配布対象の OS のアーキテクチャに合わせる) 2 . todo アプリケーションのディレクトリを electron/resources/app にコピーします。

```
$ cd todos
$ cp -r . ~/Downloads/electron-v0.26.1-win32-x64/resources/app
```

3 . アプリケーション名を todos.exe にリネームします

```
$  mv ~/Downloads/electron-v0.26.1-win32-x64/electron.exe ~/Downloads/ele
ctron-v0.26.1-win32-x64/todos.exe
```

4 . electron のディレクトリ名をリネームして zip 化します

```
$ cd ~/Downloads/
$ mv electron-v0.26.1-win32-x64 todos
$ zip -r todos todos
```

5 . 作成された todos.zip を配布します

ソースコードを隠して配布したい場合

ソースコードを隠したい場合は、asar で圧縮してから配布します。 詳しくはこちらを参照してください。

まとめ

今回は、最新の Web 技術を活用した開発フレームワーク「Electron」を調査・試用しました。 実際に使った感想として、ビルドの時間も比較的短く効率的に開発ができました。 今回は十分な説明ができませんが、Electron を活用することにより、Web アプリでは出来ないこと(ファイル操作、Shell コマンド実行、通知、ネイティブメニュー等)も実現できます。 Electron の開発も活発に進んでいる様子なので、今後も注目です。

参考

- Quick start
- Application packaging
- Electron でアプリケーションを作ってみよう