第2章 JavaScriptなどのウェブ技術とNode.jsを使ってデスクトップ・アプリが作れるフレームワーク

# Electronアプリケーションの開発環境を構築する

竹本 義孝

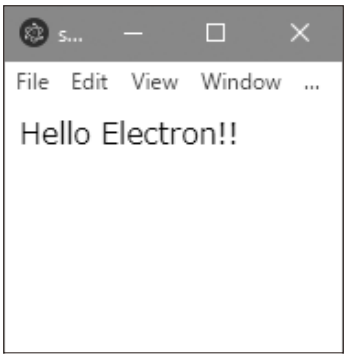


図1 実際にElectronを起動して「Hello Electron!!」プロジェクトを実行した様子
ウィンドウに「Hello Electron!!」の文字が表示されている

本章では，Blocklyアプリケーションの作成に使うフレームワークElectronについて解説します．実際に**図1**に示すようなサンプル・アプリケーションを作成しながら，Blocklyアプリケーションを開発するための環境を構築していきます．（編集部）

## ● Electronとネイティブ・ライブラリ

### ▶今回構築する環境の要件

本章では最終的にマイコンとシリアル通信を行うアプリケーションを開発します．

Electronは，Chromiumを使っているので，許可すればWeb APIのSerialPortが使えます．本稿では，Web APIのSerialPortを使って通信しますが，ポート選択の際にプロダクトIDとベンダIDが要求されます．逆に筆者が見慣れている「COM1」などを表示する手段がWeb APIにはありません．そこで，本稿ではポート一覧の取得にWeb APIではなく，Node Serialportを使います．

Node Serialportは，シリアル通信を行うためのNode.jsライブラリです．このライブラリにはC++で実装されたネイティブ・ライブラリ（@serialport/Bindings）が存在し，Node.jsのABI（Application Binary Interface）に合わせてコンパイルされます．

### ▶「Electron Builder」でビルドする

ElectronのABIは，Node.jsのABIと異なるので，単純にネイティブ・コードを使用するライブラリを使うと，実行時にエラーが発生します．

従って，Electronで開発する場合は，ライブラリをインストールするときに，ElectronのABIに合わせてコンパイルする必要があります．本稿では，Electron Builderというライブラリを使ってビルドします．

### ▶webpackを使う場合は除外する

また，Node Serialportでは共有ライブラリの検索のために，JavaScriptのファイル・パスから共有ライブラリを取得する処理が書かれています．webpackなどを用いて1つのファイルにまとめてしまうと，この動作が破壊されてしまいます．そのため，ライブラリを除外する設定が必要です．

Web APIのSerial Portだけを使う場合など，ネイティブ・ライブラリを含むライブラリを利用しないなら，特にこれらのことを気にする必要はありません．

## 1 初めてのElectronプロジェクト「Hello Electron!!」を作ってみる

まずはNode Serialportが使えるシンプルなElectronアプリケーションを作ってみます．サンプル・ディレクトリ名は`00-electron`です．

## ● ステップ1：プロジェクトの初期化

ターミナルを開いて，次のコマンドを実行し，プロジェクトを初期化します．

```
mkdir 00-electron↵
cd 00-electron↵
yarn init -y↵
yarn add -D electron
            electron-builder↵
```

## ● ステップ2：package.jsonの編集

次に，`00-electron`ディレクトリ内の`package.json`を**リスト1**の通りに編集します．編集内容は次の通りです．

1. `main`に`main.js`を指定
2. `script.postinstall`に`electron-`