

Table of Contents

Introduction	1.1
CHIRIMENとは？	1.2
B2G OSとは？	1.3
CHIRIMENの準備	1.4
WebIDE の使い方	1.5
Hello World の作り方	1.6
デバッグのやり方	1.7
adb コマンド	1.8
adb のセットアップ	1.9

chirimen-doc

this is CHIRIMEN Open Hardware Learning document making by gitbook-starter-kit

gitbook-starter-kit

GitBookのスターターキット。

以下のスライドでこの構成がどのように使われているかを解説しています。

- [Markdownで書く電子書籍開発環境](#)

実際にこの構成を利用しているプロジェクトとして以下のものがあります。

- [azu/JavaScript-Plugin-Architecture](#)

インストール

GitBookを使った書籍を以下のようにするだけで書き始めることができます。

```
git clone https://github.com/azu/gitbook-starter-kit.git your-book-name
cd your-book-name
npm install
# 必要に応じて不要な.git履歴を削除
rm -rf .git/
git init
```

使い方

```
npm start
```

GitBookのローカルサーバが立ち上がり、<http://localhost:4000/> にアクセスすることでプレビューできます。

```
npm run build
```

単純にビルドだけをしたい場合は、`npm run build` で行うことができます。（`_book/` にHTMLファイルが生成されます。）

文章を追加する

gitbook-starter-kitでは以下のようなディレクトリ構造になっています。

```
.
├── README.md
├── SUMMARY.md <= 目次
├── ja/ <= .mdの文章を追加する
├── prh.yml
├── src/ <= サンプルコード
└── test/ <= サンプルコードのテスト
```

文章を追加する `ja/` というディレクトリ名には独別な意味はないため好きな名前に変更して問題ありません。

文章を追加する場合は以下の手順で行うことができます。

1. `ja/` 以下にMarkdownファイルで文章を追加する
2. `SUMMARY.md` に追加したMarkdownファイルへのリンクを書く

追加した後は `npm start` などでGitBookでプレビューすれば表示を確認することができます。(デフォルトで自動的にリロードされるようになっています。)

テスト

```
npm test
```

`npm test`で以下のテストが実行されます。

- [ESLint](#)でのコードチェック
- [textlint](#)での文章チェック
- [Mocha](#)でのサンプルコードのテスト

並列でテストを実行できるように[npm-run-all](#)を利用しています。テスト結果の表示が混ざるのが気になる場合は、`--parallel` オプションを外してみてください。

textlint

[textlint](#)を使った文章のチェックが行われます。デフォルトでは技術書向けの設定が導入されているため目的にあわせてtextlintのルールを設定してください。

- [textlintで日本語の文章をチェックする | Web Scratch](#)
- [Collection of textlint rule · textlint/textlint Wiki](#)
- [textlint-jp/textlint-rule-preset-jp-technical-writing: 技術文書向けのtextlintルールプリセット](#)

表記揺れ

[prh.yml](#)に辞書を追加することで表記揺れをチェックすることができます。詳しい設定方法については以下を参照してください。

- [textlint + prhで表記ゆれを検出する | Web Scratch](#)

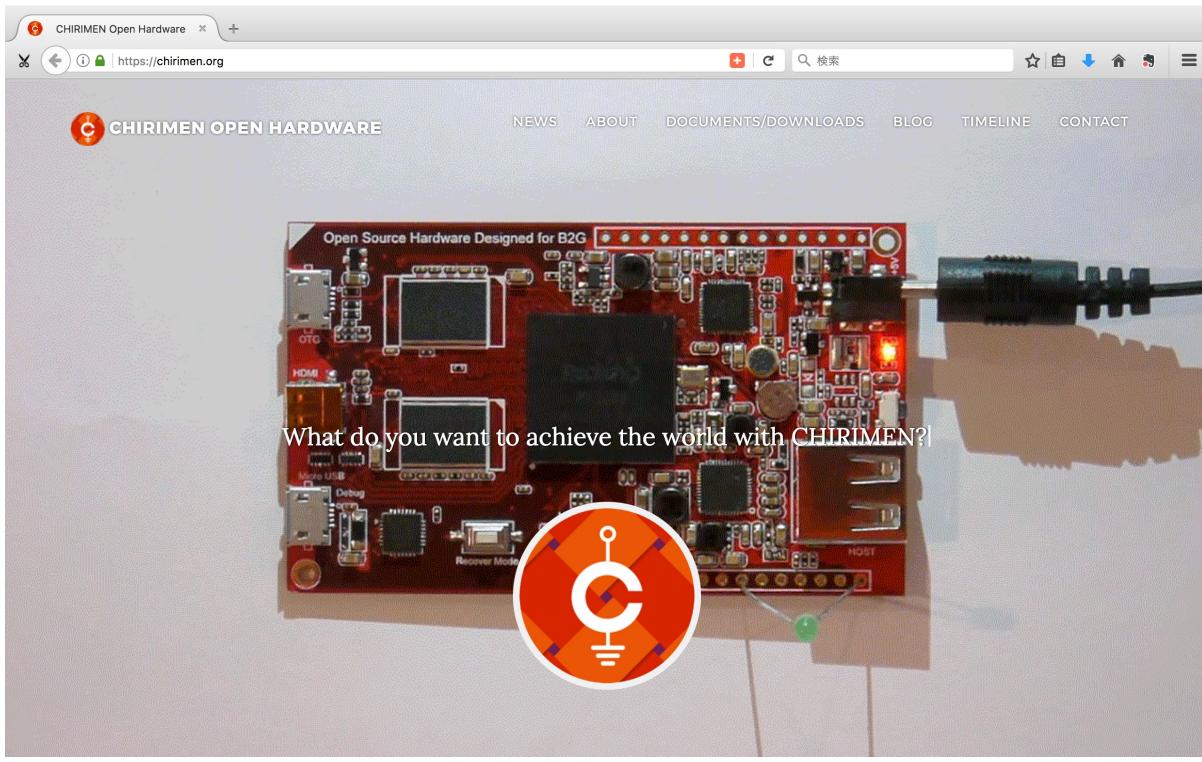
Contributing

1. Fork it!
2. Create your feature branch: `git checkout -b my-new-feature`
3. Commit your changes: `git commit -am 'Add some feature'`
4. Push to the branch: `git push origin my-new-feature`
5. Submit a pull request :D

License

MIT

CHIRIMENとは



The screenshot shows the CHIRIMEN Open Hardware website. At the top, there's a navigation bar with links for NEWS, ABOUT, DOCUMENTS/DOWNLOADS, BLOG, TIMELINE, and CONTACT. Below the navigation is a large image of a red BeagleBoard (Revision B2G) with various components like a central processor, memory chips, and connectors. A USB cable is connected to one of the ports, and a green LED is illuminated. Overlaid on the board is a large orange circular logo with a stylized 'C' and a central node-like symbol. The text "What do you want to achieve the world with CHIRIMEN?" is written in white across the center of the board. Below the main image, there's a text box containing the following information:

CHIRIMEN はセンサーやアクチュエーターなどの物理デバイスをWebブラウザ技術だけで制御することができる開発環境で、ボードコンピュータとその上で動作するソフトウェアを含めた総称です。

ボードコンピュータとしてのハードウェア、その上で動作するウェブブラウザ ※1 ソフトウェア、センサーや物理デバイスをウェブアプリ・JavaScript から制御するためのWebGPIO やWebI2C といった低レベルデバイス API の実装などが含まれており、CHIRIMEN Open Hardwareコミュニティによって開発され、CHIRIMEN というコードネームがつけられました。

CHIRIMEN に関連するハードウェアとソフトウェアのソースコードは、オープンソースとして公開されます。

※1: 現在は Boot to Gecko を使用

引用

- [CHIRIMEN FAQ](#) より引用

B2G OSとは

B2G OS の用語

- B2G OS のドキュメントをさらに読み進める前に理解しておくべき用語がいくつかあります。

B2G

- Boot to Gecko の略語です。

Gecko

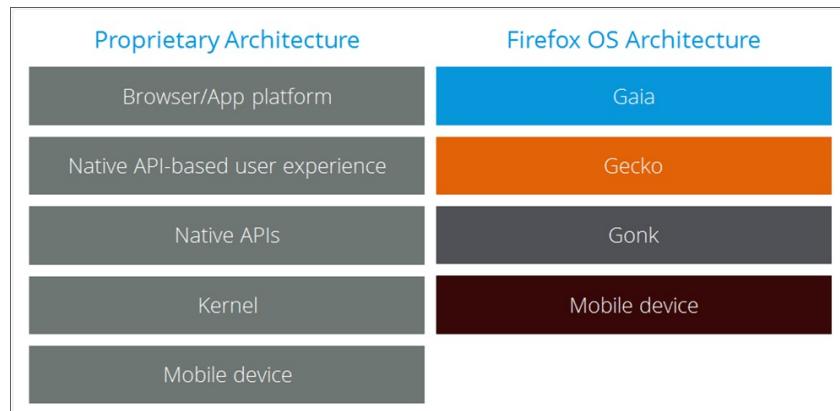
- Gecko は、Mozilla プロジェクトによって開発されているレイアウトエンジンの名称です。

Q. Boot to Gecko とは何ですか？

B2G OS は、コミュニティが保守するスマートフォンやタブレット、スマート TV、コネクテッドデバイス向けのオープンソースのオペレーティングシステムです。

アーキテクチャの全体像

以下の図は、プロプライエタリなプラットフォームと B2G OS のアーキテクチャを比較したものです。



引用

- [B2G OS のアーキテクチャ](#) より引用
- [B2G](#) より引用
- [Gecko](#) より引用
- [CHIRIMEN FAQ](#) より引用

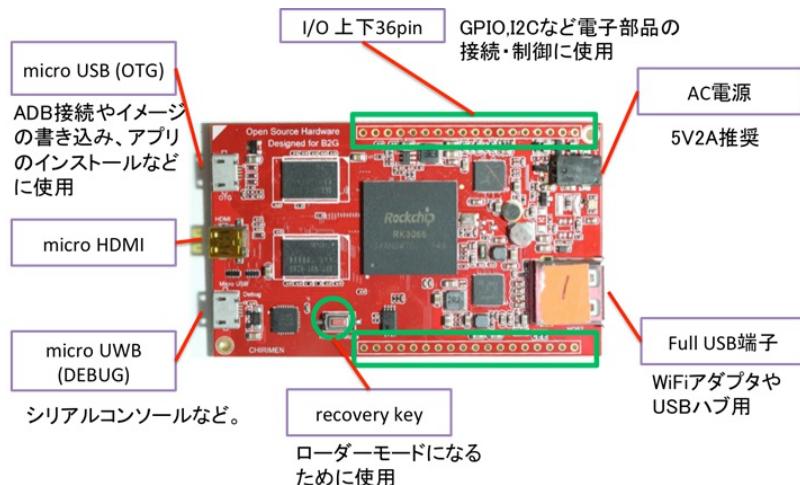
CHIRIMENの準備

準備するもの

- CHIRIMEN
- USB-microUSBケーブル
※通常CHIRIMENに同梱されています。
CHIRIMEN-PCを接続するため、データ通信用(充電のみ)用ではダメです。
- microHDMI-HDMIケーブル
※通常CHIRIMENに同梱されています。
CHIRIMEN-モニタを接続するため
- CHIRIMEN用電源
※通常CHIRIMENには電源ケーブル(USB-DC)が同梱されていますが、加えて
USB Aで5V 1A以上(2A以上推奨)を給電可能な電源が別途必要となります。
- HDMIディスプレイ
- USBマウス

CHIRIMEN の構成(起動する前に読んで)

- CHIRIMENのインターフェースはこんな感じです。



CHIRIMEN の接続

- 下図②microHDMI端子にディスプレイを接続します。
- 下図⑤Full USB端子にマウスを接続します。
下図ではUSB HUBを用いていますが、直接さすことも可能です。
- 上図micro USB(OTG)端子にPCを接続します。
- 下図④CHIRIMEN用電源ケーブルを接続します。
- CHIRIMEN上のLEDが点灯し、しばらく経つとB2Gが起動する画面が現れます。
 - [Echigo rev.1 New firmware CMN2015-1_B2GOS-20170301](#)
- アプリ画面が表示されたら、マウスを使用して操作をしてみましょう。
左クリックが選択、右クリックがホームボタンの役割となります

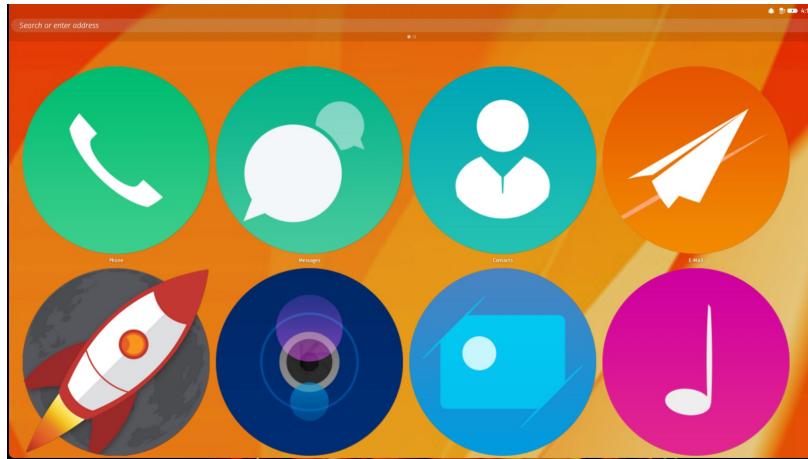


正しく起動しない場合

- 絵の通り配線はきちんとされているでしょうか？
- 電源ケーブルが抜けかかっている、HDMIケーブルがきちんとささっていないなどはないでしょうか？
- 物理的に正しくつながっていないケースが多いのでもう一度見直してみてください。
- 上記以外は、物理的にPCが認識しているかを確認する事が必要です。
PCが正しく認識しているか確かめる方法は、[adb コマンド](#) を用いて確認します。
- [adb コマンド](#) を用いても認識されない場合
 - 物理的に破損している可能性があります。
 - OSイメージがボードに正しく焼かれていない可能性があります。

正しく起動した場合

- B2G OSが表示されます。



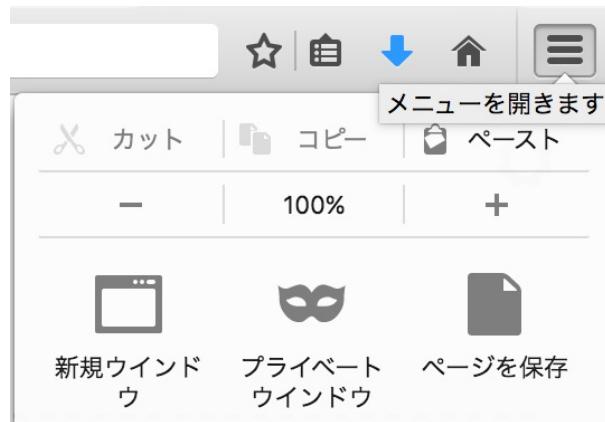
引用

- [CHIRIMEN Hello world](#) より引用
- [ボードコンピュータ コネクタ配置](#) より引用

WebIDE の使い方

WebIDE の出し方

- バーガーボタンをクリックします。



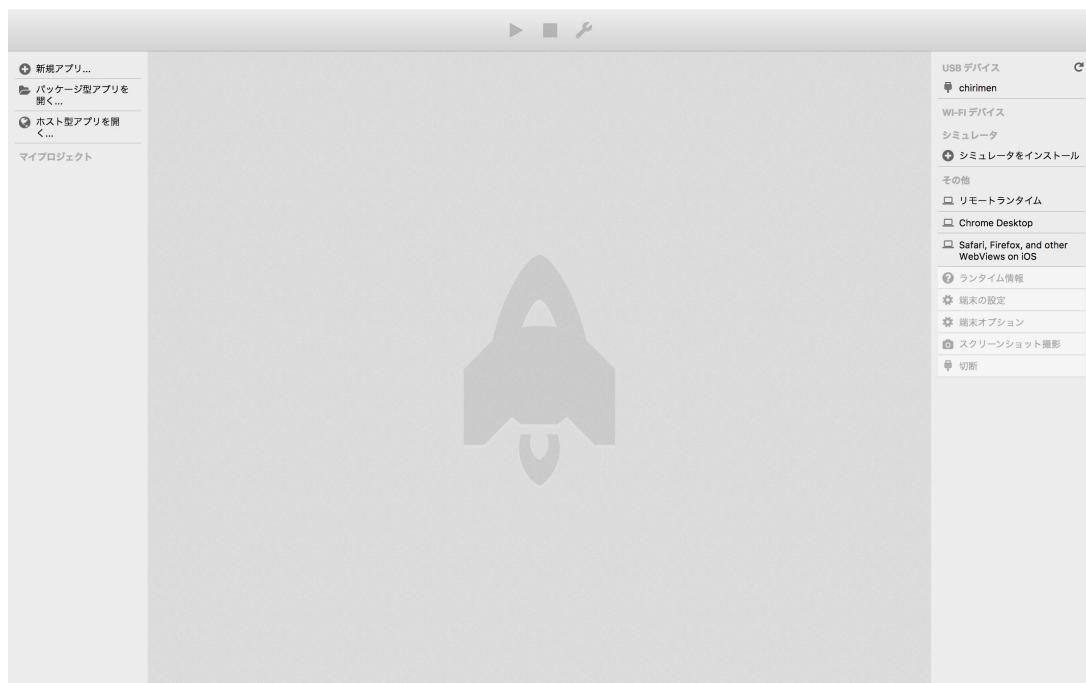
- 開発ツールをクリックします。



- WebIDEをクリックします。



- WebIDE が表示されました。



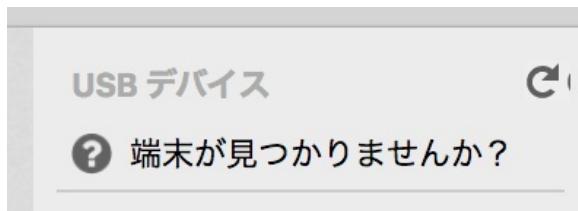
CHIRIMENの接続確認のやり方

- 下図のように「chirimen」と表示されていれば、接続OKです。



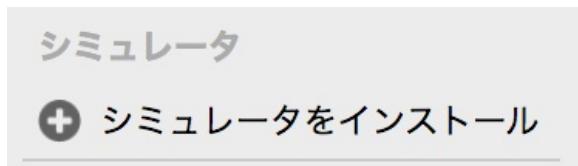
- 表示されていない場合は、adb コマンドを用いて確認します。

※接続又は認識されていない表示例



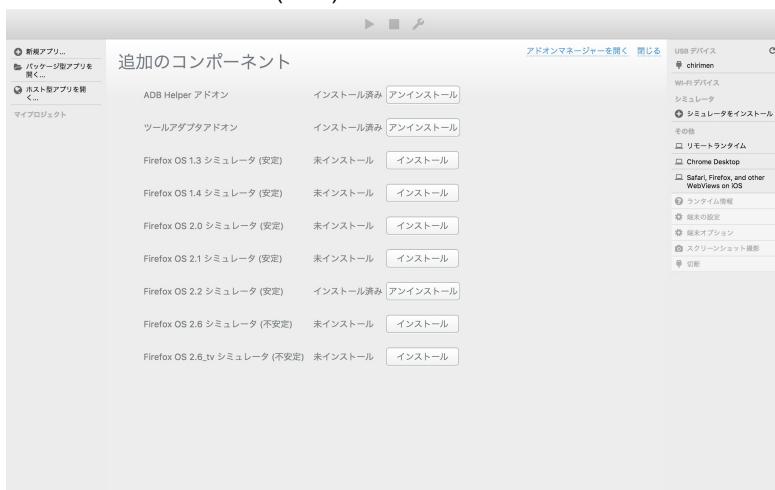
アドオンインストールの確認のやり方

- 「シミュレータをインストール」をクリックします。



- 下記コンポーネントがインストールされている事を確認します。

- ADB Helper アドオン(必須)
- ツールアダプタアドオン(必須)
- Firefox OS x.x シミュレータ(任意)



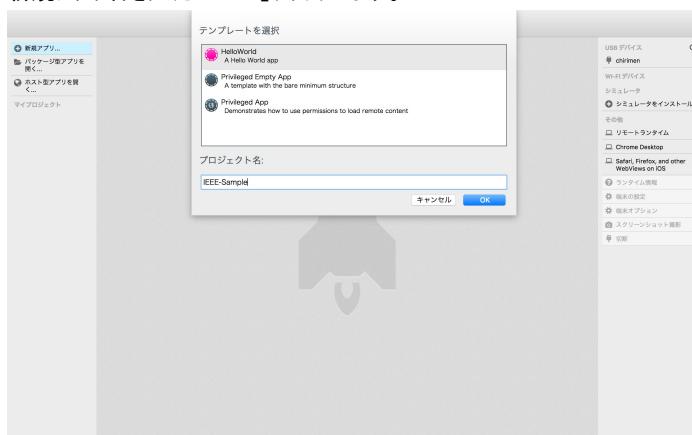
Hello World の作り方

Hello World の作成の手順

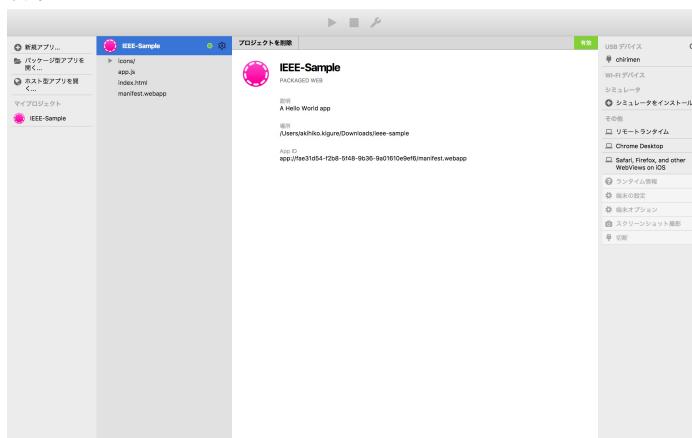
- 新規アプリをクリックします。



- 新規アプリ名を入力し「OK」クリックします。



- 新規アプリ名のプロジェクトが作成されました。

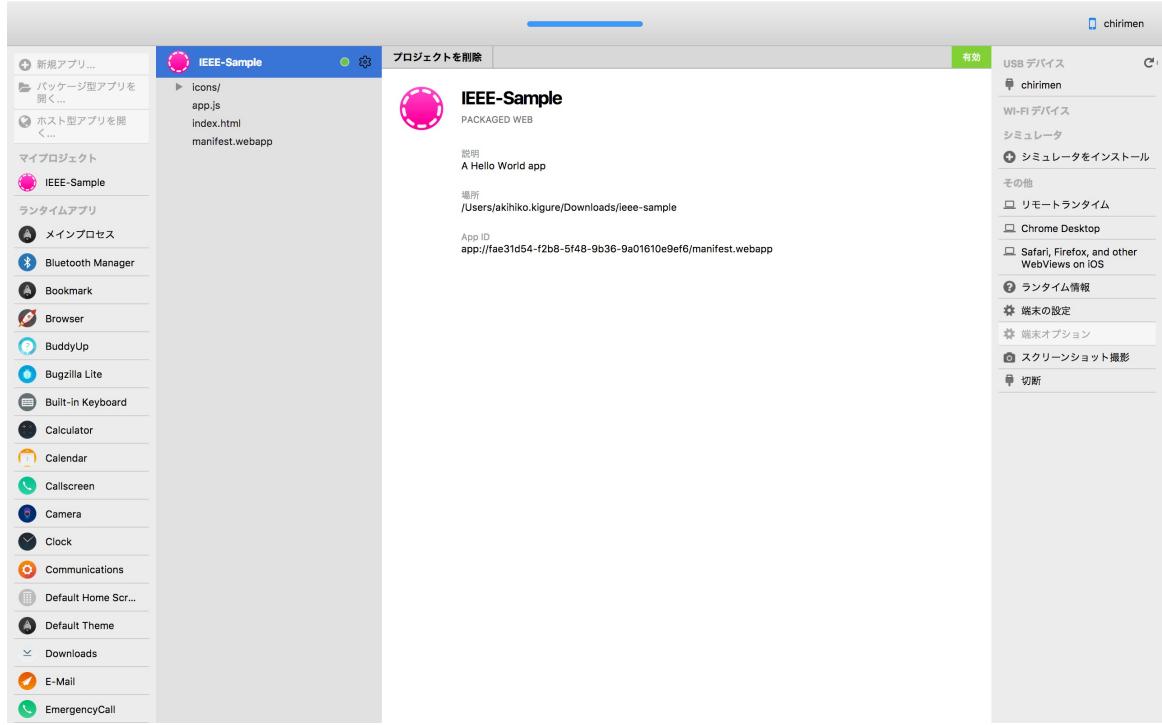


Hello World のインストールのやり方

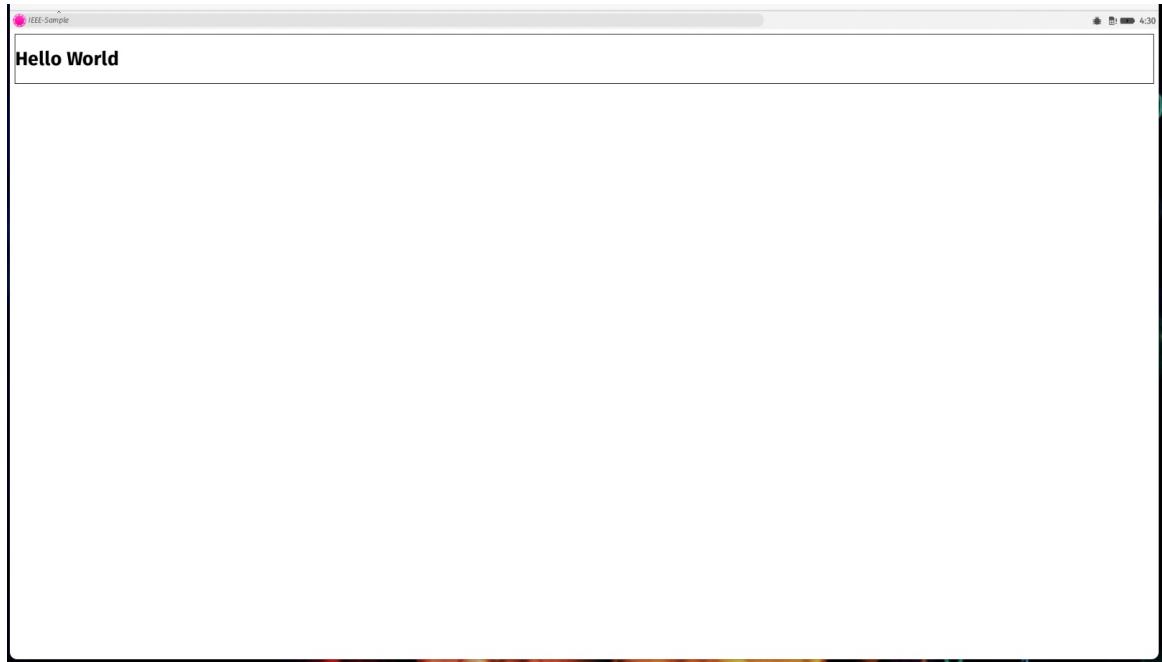
- USBデバイスに表示されている「chirimen」をクリックします。
- △マークの実行ボタンが押せるようになります。



- △マークの実行ボタンをクリックし、アプリケーションをインストールします。



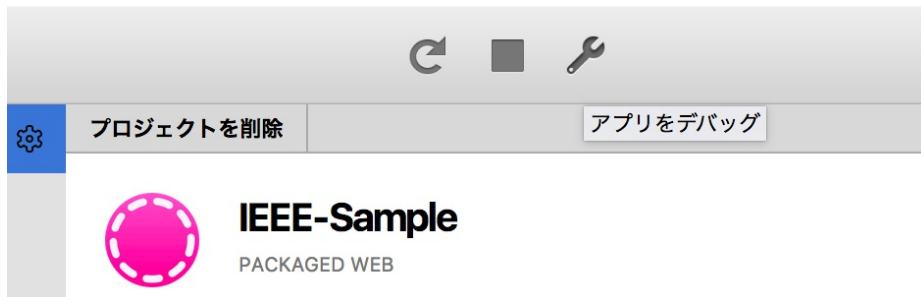
- アプリケーションが起動するとCHIRIMMENに接続したディスプレイが下図のようになります。



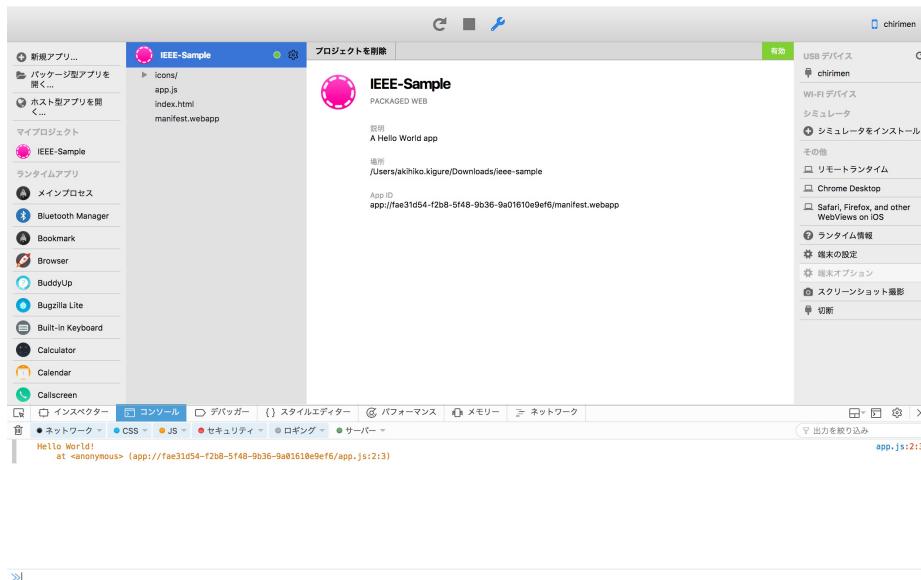
デバッグのやり方

デバッグ画面表示手順

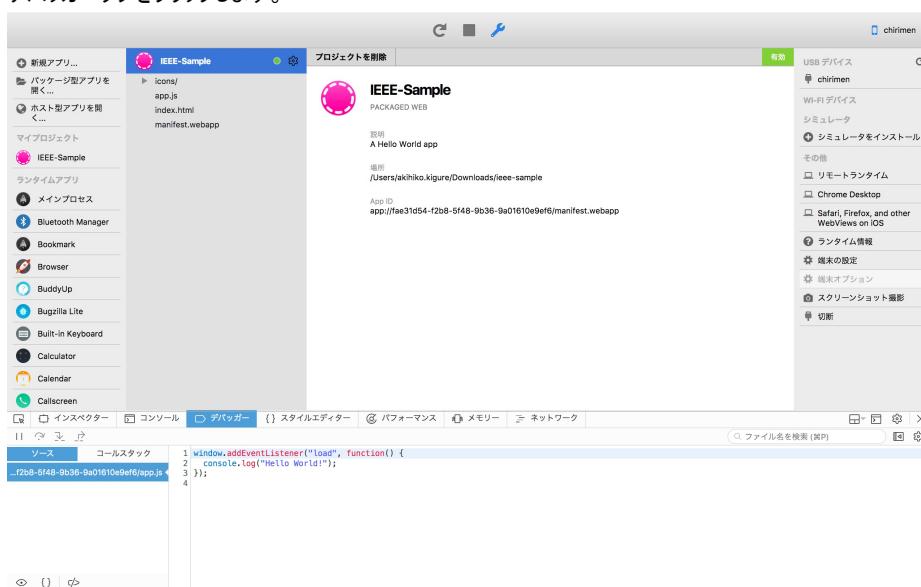
- レンチマークのアイコンをクリックします。



- WebIDEウィンドウ下部デバッグ画面が表示されます。



- デバッガータブをクリックします。



デバッグの手順

- 任意の行にブレークポイントを設定します。

行番号をクリックすると下図のようになります。



The screenshot shows the developer tools interface with the "デバッガ" (Debugger) tab selected. The code editor displays a script with four lines of code. Line 2, which contains the statement `console.log("Hello World!");`, has a small blue square icon indicating a breakpoint is set there.

```
1 window.addEventListener("load", function() {  
2   console.log("Hello World!");  
3 });  
4
```

- 矢印マークの実行ボタンクリックします。



- 設定されたブレークポイントでプログラムが一時停止します。



The screenshot shows the developer tools interface with the "デバッガ" (Debugger) tab selected. The code editor displays the same script as before, but now the second line, which contains the `console.log` statement, is highlighted in green, indicating the program is paused at that point. The status bar at the bottom of the browser window also shows "(anonymous) app://...app.js:2".

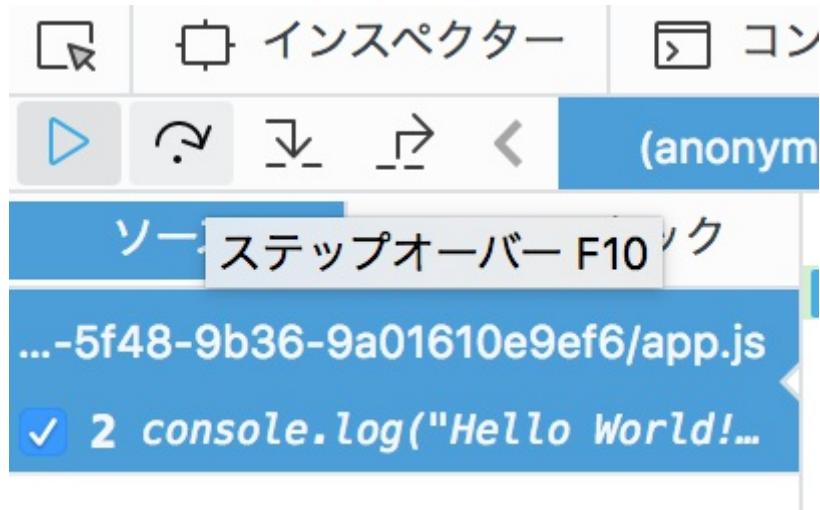
```
1 window.addEventListener("load", function() {  
2   console.log("Hello World!");  
3 });  
4
```

デバッグの進め方の種類

- 復帰(F8)



- ステップオーバー(F10)



- ステップイン(F11)



- ステップアウト(↑+F11)



adb のコマンドの使い方

デバイス確認

- CHIRIMENボードがハードウェア的に接続されているかを確認します。

```
→ ~ adb devices
List of devices attached
* daemon not running. starting it now at tcp:5037 *
* daemon started successfully *
0123456789ABCDEF    device  <= 正常に接続している例

→ ~ adb devices
List of devices attached
* daemon not running. starting it now at tcp:5037 *
* daemon started successfully *

→ ~ <= 接続が確認できないと何も表示されない
```

adb 起動

- adb プロセスを起動します。

```
→ ~ adb start-server
* daemon not running. starting it now at tcp:5037 *
* daemon started successfully *
→ ~
```

adb 停止

- adb プロセスを停止します。

```
→ ~ adb kill-server
→ ~
```

adb 再起動

- adb プロセスを再起動します。

```
→ ~ adb reboot
→ ~
```

adb シエル

- CHIRIMEN内で、Linuxコマンド操作をする時に遣います。

```
→ ~ adb shell
shell@chirimene:/ $ <= 正常にEchigo Rev.1にログイン
shell@chirimene:/ $ exit
→ ~ <= 元のターミナルに戻る
```

```
→ ~ adb shell  
error: no devices/emulators found <= 接続が確認できないと表示されます。  
→ ~
```

adb devices で、認識しない時の対応

- USBケーブルがPCに接続されている事を確認します。
- ボードがOS起動中かどうかをモニターを見て判断します。

上記以外で認識しない場合

- adb_usb.ini 編集します。
 - ディレクトリを作成・移動します。

```
$ mkdir ~/.android  
$ cd ~/.android  
$ vi ~/.android/adb_usb.ini
```

- 下記の内容を追記します。

```
# Echigo Rev.1  
# 1 USB VENDOR ID PER LINE.  
0x2207
```

- 再度、adb devices を実行します。

引用

- [Android Debug Bridge](#) より引用

adb のセットアップ

android-tools-adbをインストールします。

- Ubuntu 12.10以降

```
$ sudo apt-get install -y android-tools-adb
```

- Fedora 19以降

```
$ sudo yum install android-tools -y
```

- Mac OSX以降([Homebrew](#) を使っている時)

```
$ brew install android-platform-tools
```

- Windows 7 以降

- [ADBコマンド導入の方法](#)

引用

- [ADBをインストールして使用する](#) より引用
- [ADBコマンド導入の方法](#) より引用