Firefox OS アプリ開発ハンズオン

情報通信技術教育者合同会議2014　2015/03/10

Mozilla Japan テクニカルマーケティング
清水智公


Doing good is
part of our code.
mozilla.org/firefox
PORT OF OAKLAND
iPad 2

Firefox OS


検索語またはアドレスを入力
SMS
Marketplace
メール
カメラ
ソーシャル
音楽
時計
動画
音楽


午前 10:45
検索語またはアドレスを入力
電話
メッセージ
連絡先
ブラウザ
Marketplace
カメラ
ギャラリー
音楽
カレンダー
動画
環境設定
時計
メール
通信量管理

すべてWeb技術でできています！

通知、プッシュ通知 →

電話、SMS →

カメラ →

ホーム画面、壁紙 →

SDカード →

メール →

← ネットワーク、電池

← ブラウザ

← 動画、音楽

← 通信料管理

mozilla

HTML CSS JavaScript

Web アプリ

ZIP

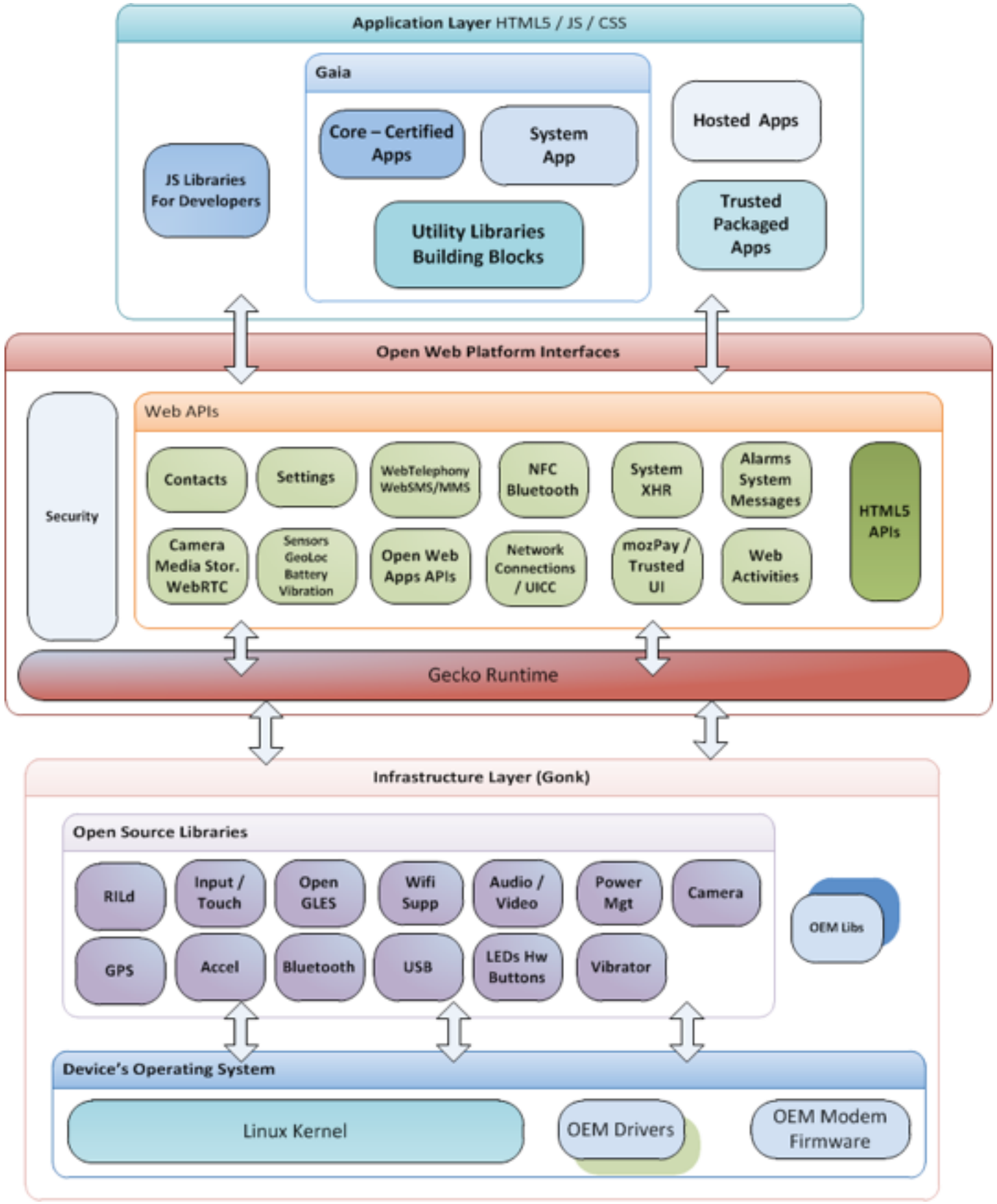
Packaged Web アプリ

Web Platform

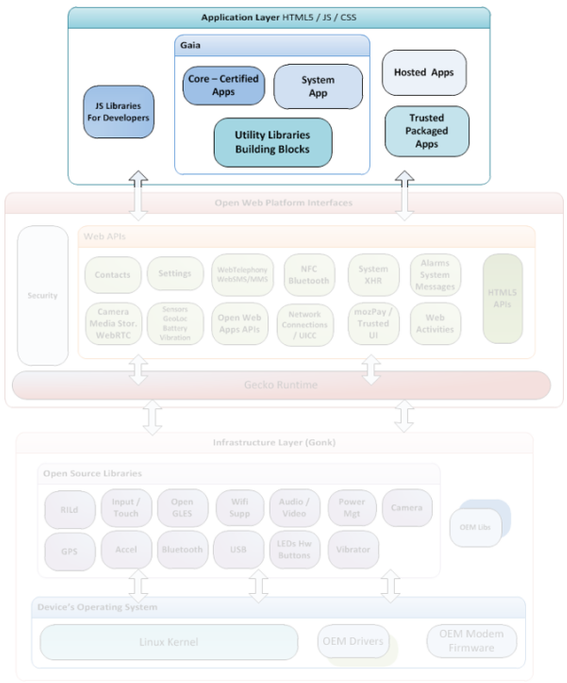
Gecko (Web Engine)

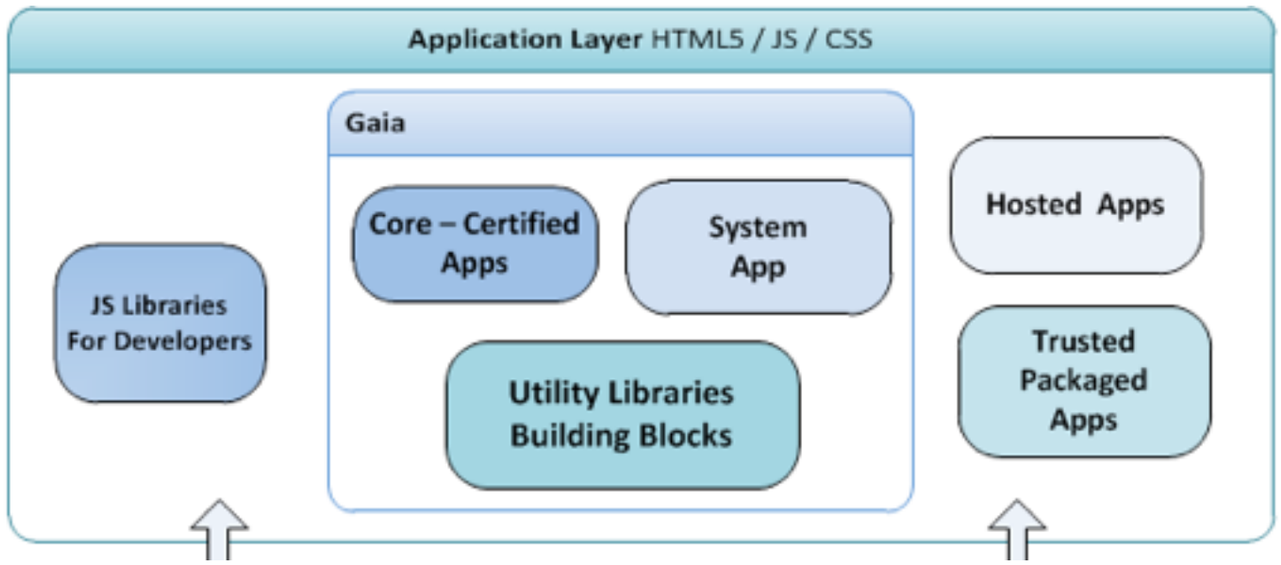
Web API

Linux カーネル、ドライバー

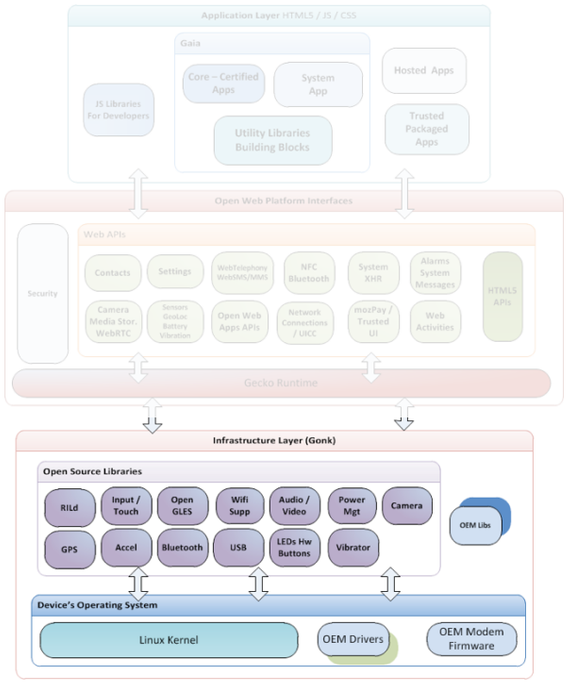

Application Layer HTML5 / JS / CSS
Gaia
Core – Certified Apps
System App
Utility Libraries Building Blocks
JS Libraries For Developers
Hosted Apps
Trusted Packaged Apps
Open Web Platform Interfaces
Security
Web APIs
Contacts
Settings
WebTelephony WebSMS/MMS
NFC Bluetooth
System XHR
Alarms System Messages
HTML5 APIs
Camera Media Stor. WebRTC
Sensors GeoLoc Battery Vibration
Open Web Apps APIs
Network Connections / UICC
mozPay / Trusted UI
Web Activities
Gecko Runtime
Infrastructure Layer (Gonk)
Open Source Libraries
RILd
Input / Touch
Open GLES
Wifi Supp
Audio / Video
Power Mgt
Camera
GPS
Accel
Bluetooth
USB
LEDs Hw Buttons
Vibrator
OEM Libs
Device's Operating System
Linux Kernel
OEM Drivers
OEM Modem Firmware

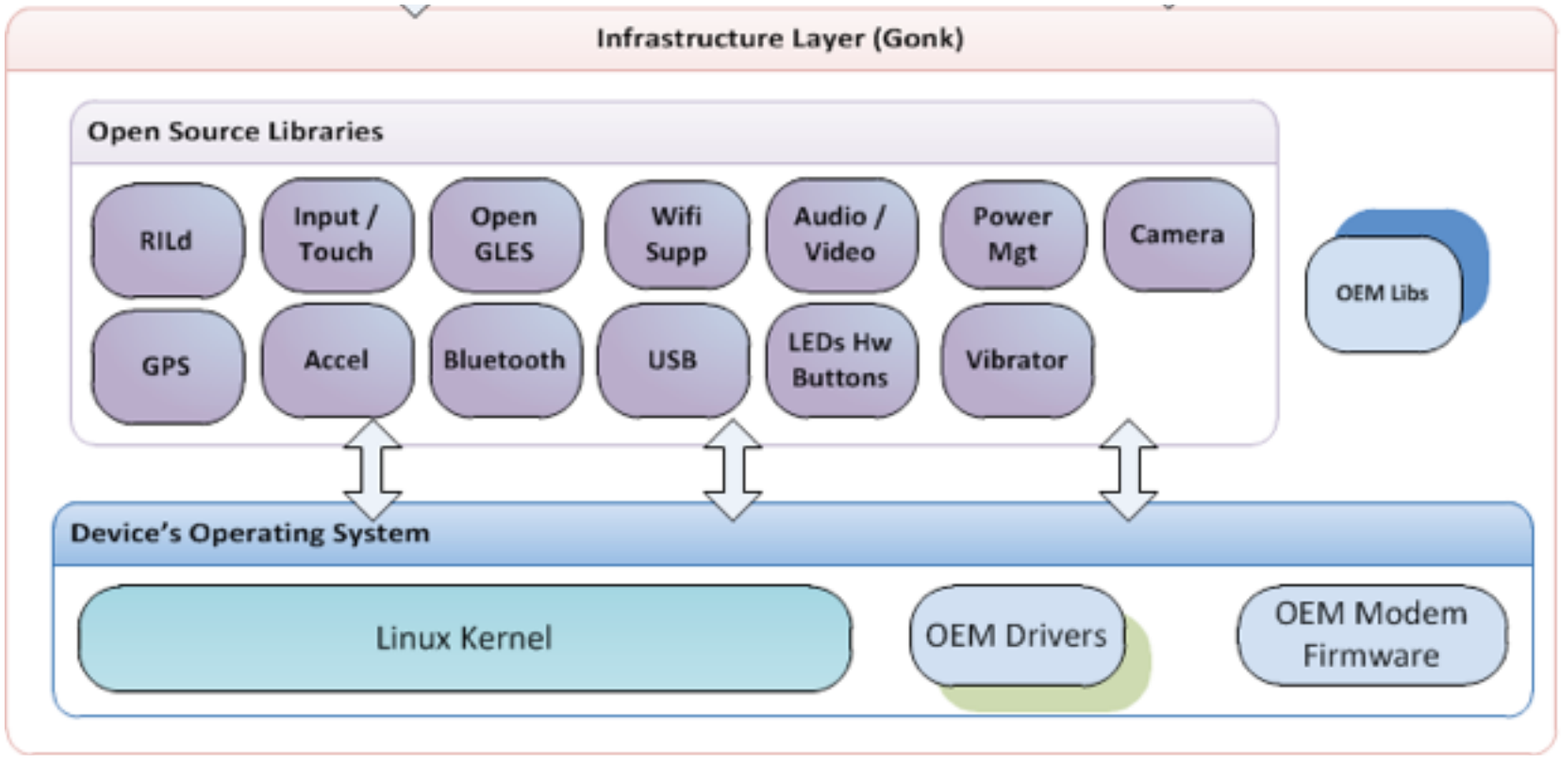
アプリケーション


Application Layer HTML5 / JS / CSS
Gaia
Core – Certified Apps
System App
Utility Libraries Building Blocks
JS Libraries For Developers
Hosted Apps
Trusted Packaged Apps
Open Web Platform Interfaces
Web APIs
Security
Contacts
Settings
WebTelephony WebSMS/MMS
NFC Bluetooth
System XHR
Alarms System Messages
HTML5 APIs
Camera Media Stor. WebRTC
Sensors GeoLoc Battery Vibration
Open Web Apps APIs
Network Connections / UICC
mozPay / Trusted UI
Web Activities
Gecko Runtime
Infrastructure Layer (Gonk)
Open Source Libraries
RILd
Input / Touch
Open GLES
Wifi Supp
Audio / Video
Power Mgt
Camera
GPS
Accel
Bluetooth
USB
LEDs Hw Buttons
Vibrator
OEM Libs
Device's Operating System
Linux Kernel
OEM Drivers
OEM Modem Firmware

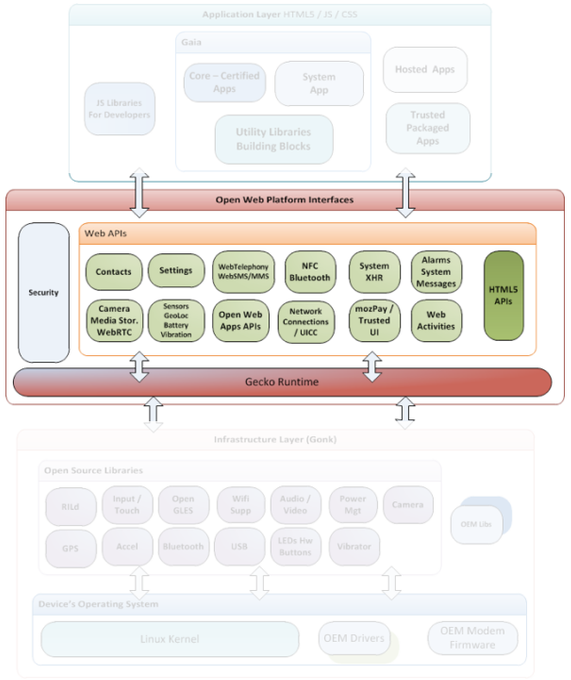

Application Layer HTML5 / JS / CSS
Gaia
Core – Certified Apps
System App
Utility Libraries Building Blocks
JS Libraries For Developers
Hosted Apps
Trusted Packaged Apps

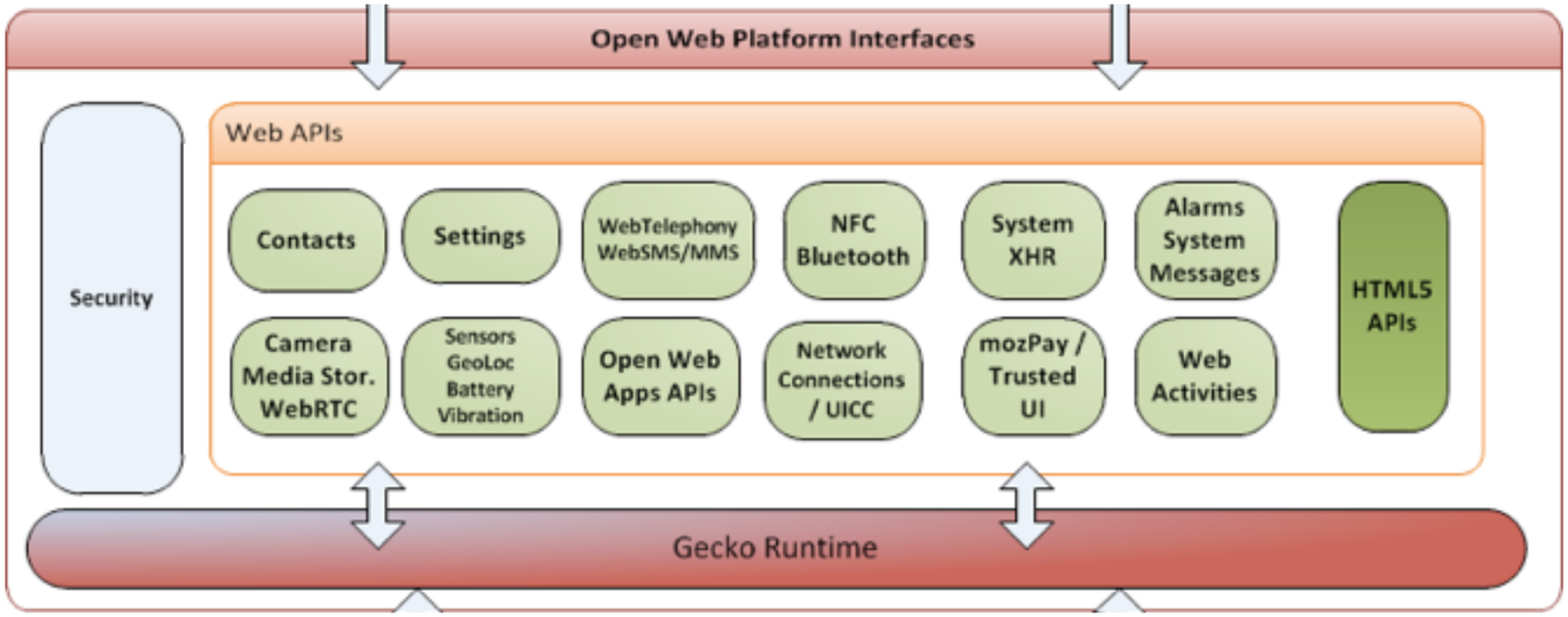
Gonk：インフラストラクチャ

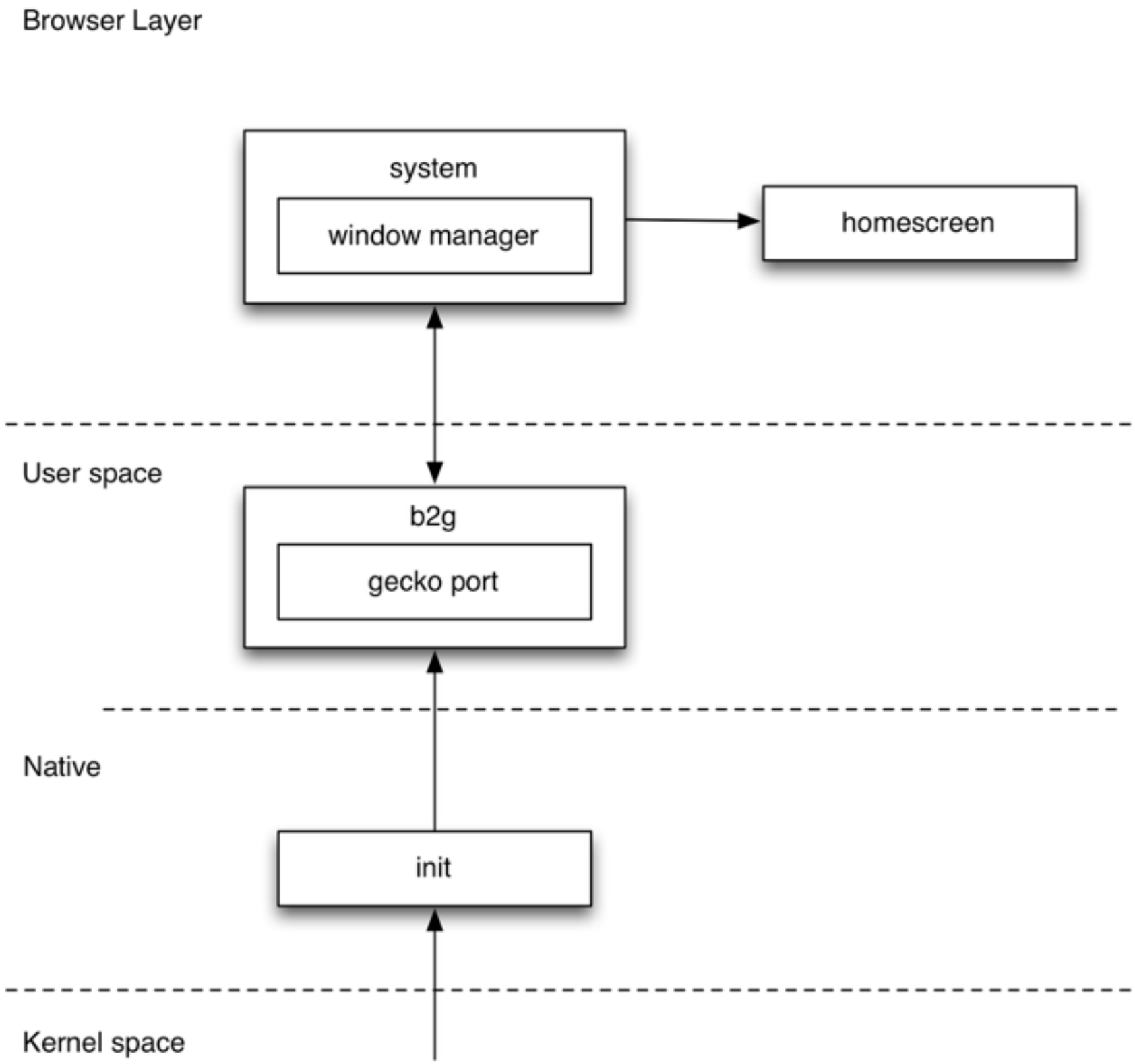




Gecko：Open Webプラットフォーム






Browser Layer
system
window manager
homescreen
User space
b2g
gecko port
Native
init
Kernel space

すべてWeb技術でできています！

通知、プッシュ通知 →

電話、SMS →

カメラ →

ホーム画面、壁紙 →

SDカード →

メール →

← ネットワーク、電池

← ブラウザ

← 動画、音楽

← 通信料管理

mozilla

詳細はMDNのアプリセンターへ！

https://developer.mozilla.org/Apps

今日の内容

- アプリ開発環境の準備、確認
- HelloWorld
  - 初めてのアプリ作成
  - センサーからの入力に合わせた変化
- カメラアプリ
  - カメラからの入力の受けつけ
  - 画像処理
  - 保存

開発環境の準備

アプリの開発を始めるためには

- 次のソフトをインストールします
  - Firefox (Developer Editionだとさらに望ましい)
  - Firefox OS シミュレータ（2.0以降）
  - 実機接続のためのドライバ（Windowsのみ）
- 必要ないこと
  - SDKのダウンロード
  - 開発者登録、登録料の支払い、規約 / 誓約書 / NDAへのサイン

Firefox®

Developer Edition

- 普通のFirefoxでも開発できますが、
  開発者が必要とする機能が充実しているため、
  こちらのほうが効率良く開発が行えます。
- https://mozilla.org/firefox/channel/
  よりダウンロードできます

mozilla

Firefox Developer Editionのインストール

https://mozilla.org/firefox/channel/ へアクセス

mozilla

Firefox Developer Editionのインストール

mozilla

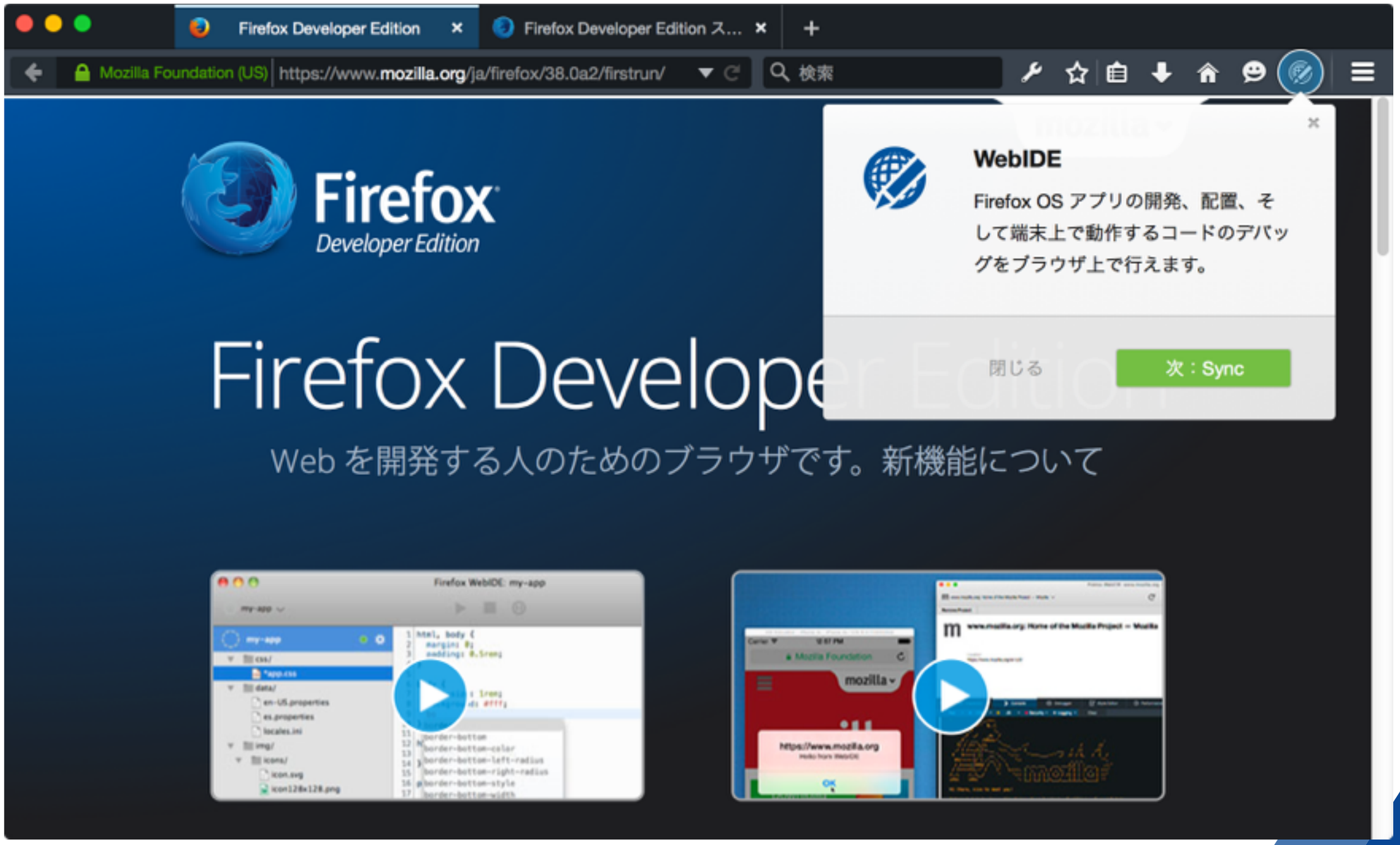
Firefox Developer Editionのインストール

ボタンをクリックして、ダウンロード開始

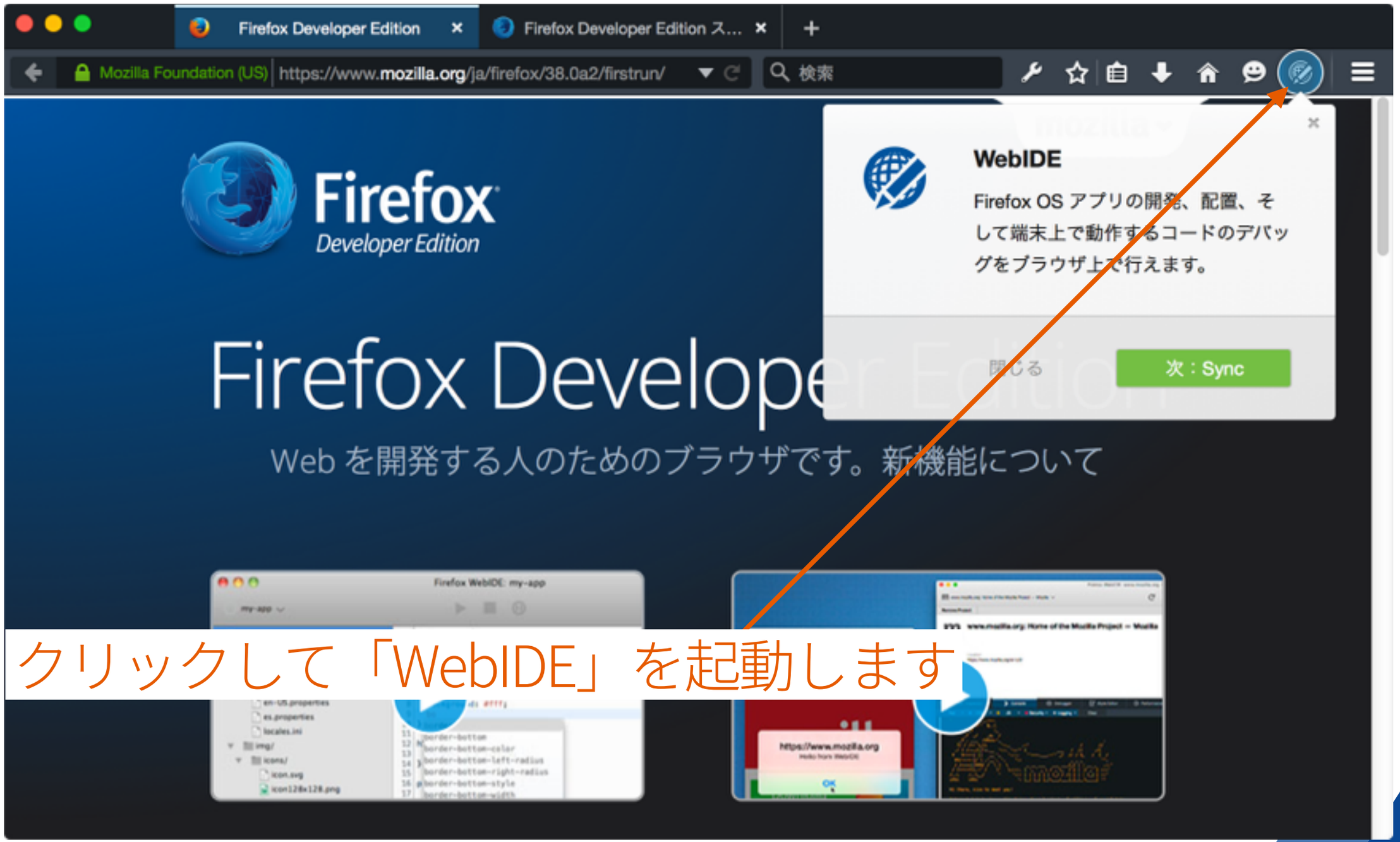
mozilla

インストールします

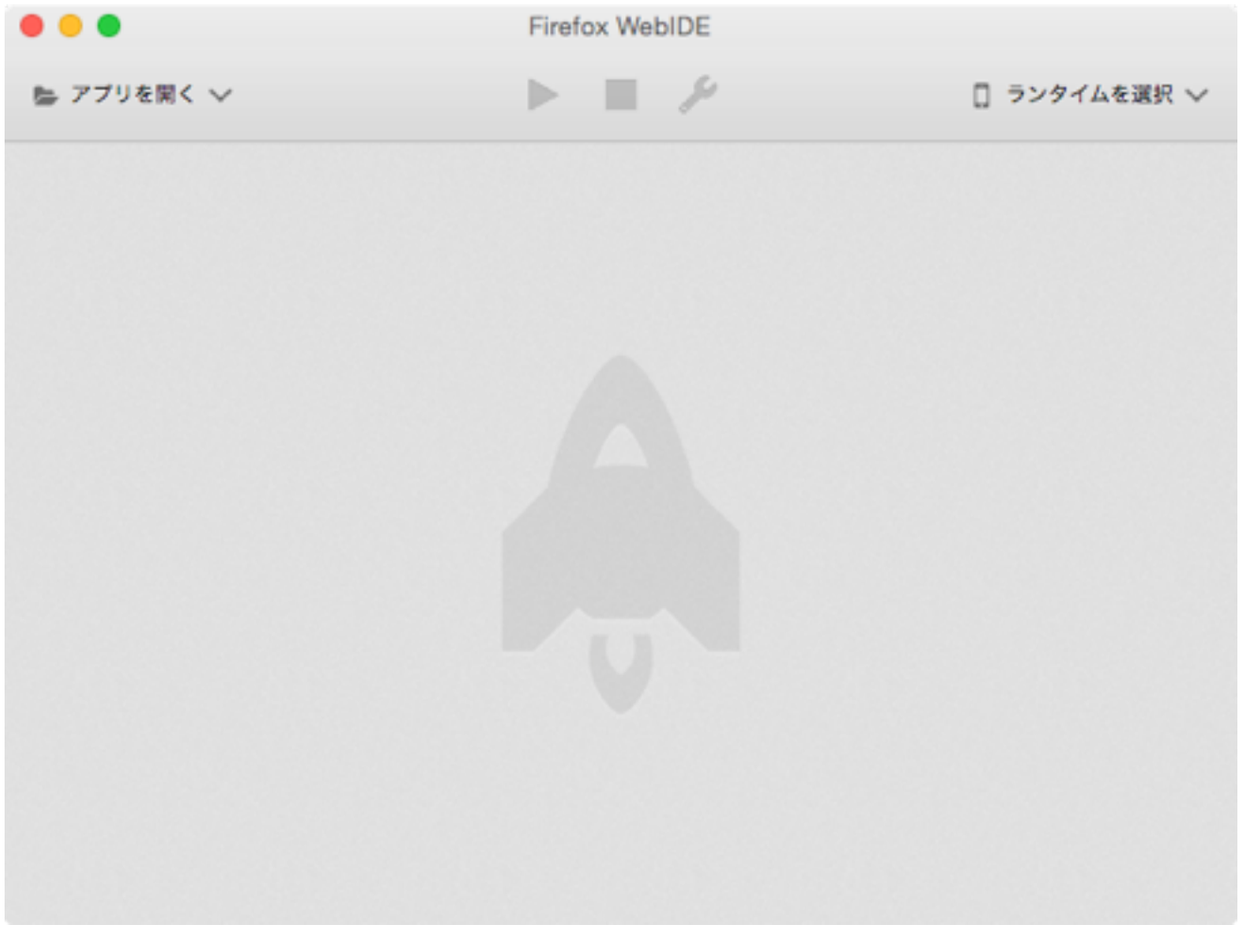
Firefox Developer Edition の起動画面



シミュレータのインストール



シミュレータのインストール（つづき）



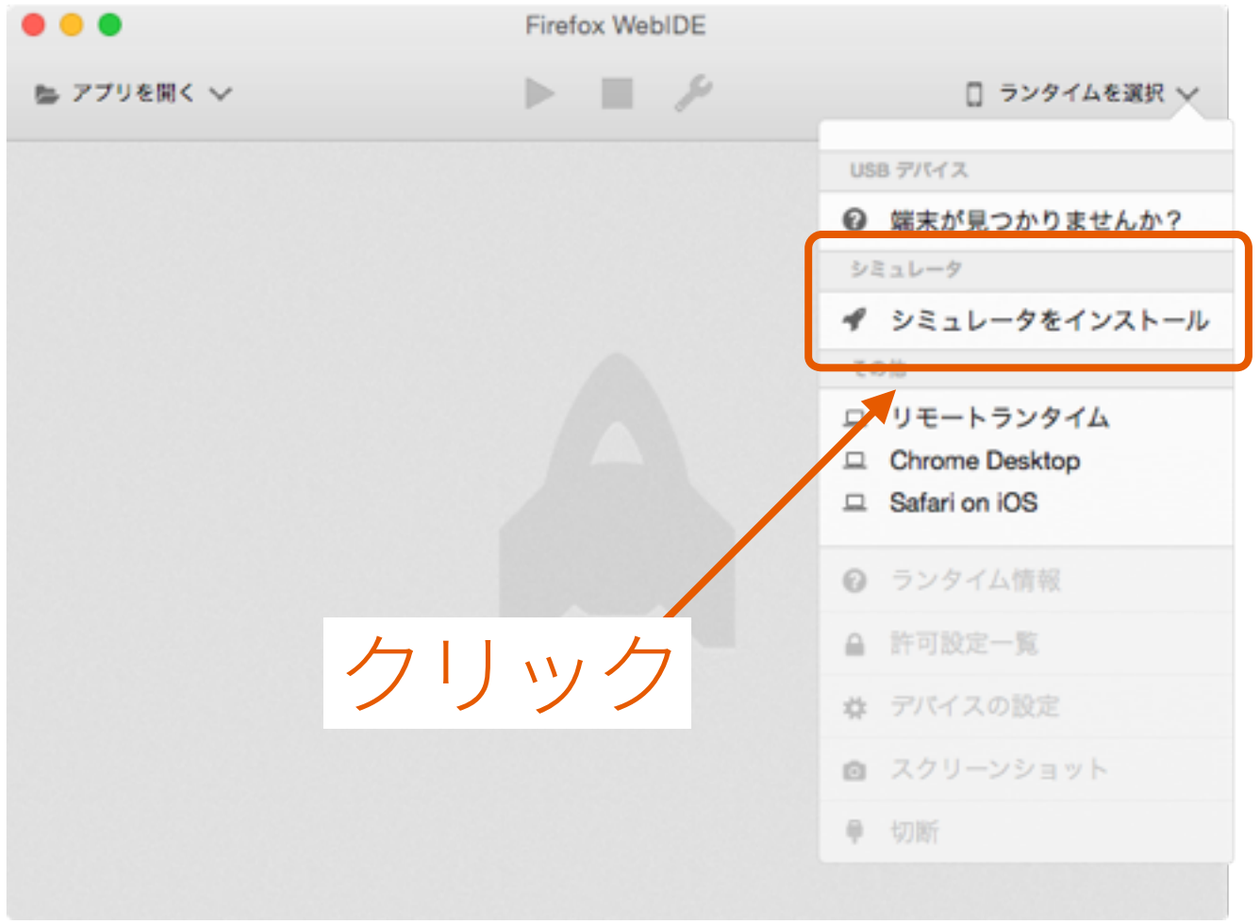
このようなウィンドウが表示されます。

詳細：https://developer.mozilla.org/docs/Tools/WebIDE

シミュレータのインストール（つづき）

「ランタイムを選択」をクリックします

シミュレータのインストール（つづき）



「シュミレータをインストール」をクリックします

mozilla

シミュレータのインストール（つづき）

クリック

「Firefox OS 2.1 シミュレータ」をインストールします

mozilla

シミュレータのインストール（つづき）

「閉じる」をクリックして終了です

実機(Firefox Flame)との接続準備

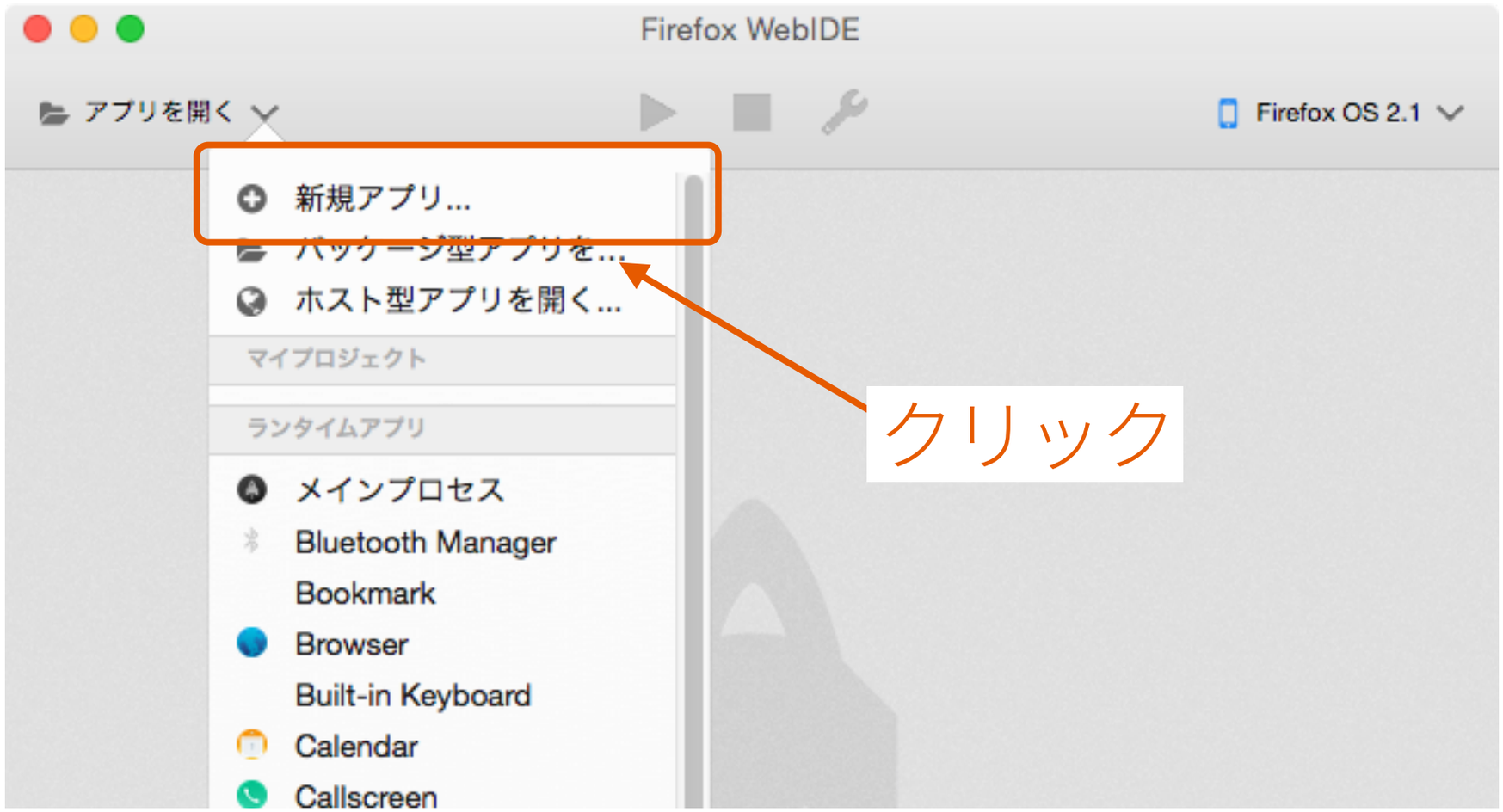
- Mac の場合：
  必要な作業はありません
- Linux の場合：
  udevルールの追加が必要です
- Windowsの場合：
  USBドライバのインストールが必要です

- 詳しくは http://goo.gl/ZJV93x を参照してください

以上で開発環境の準備は終了です

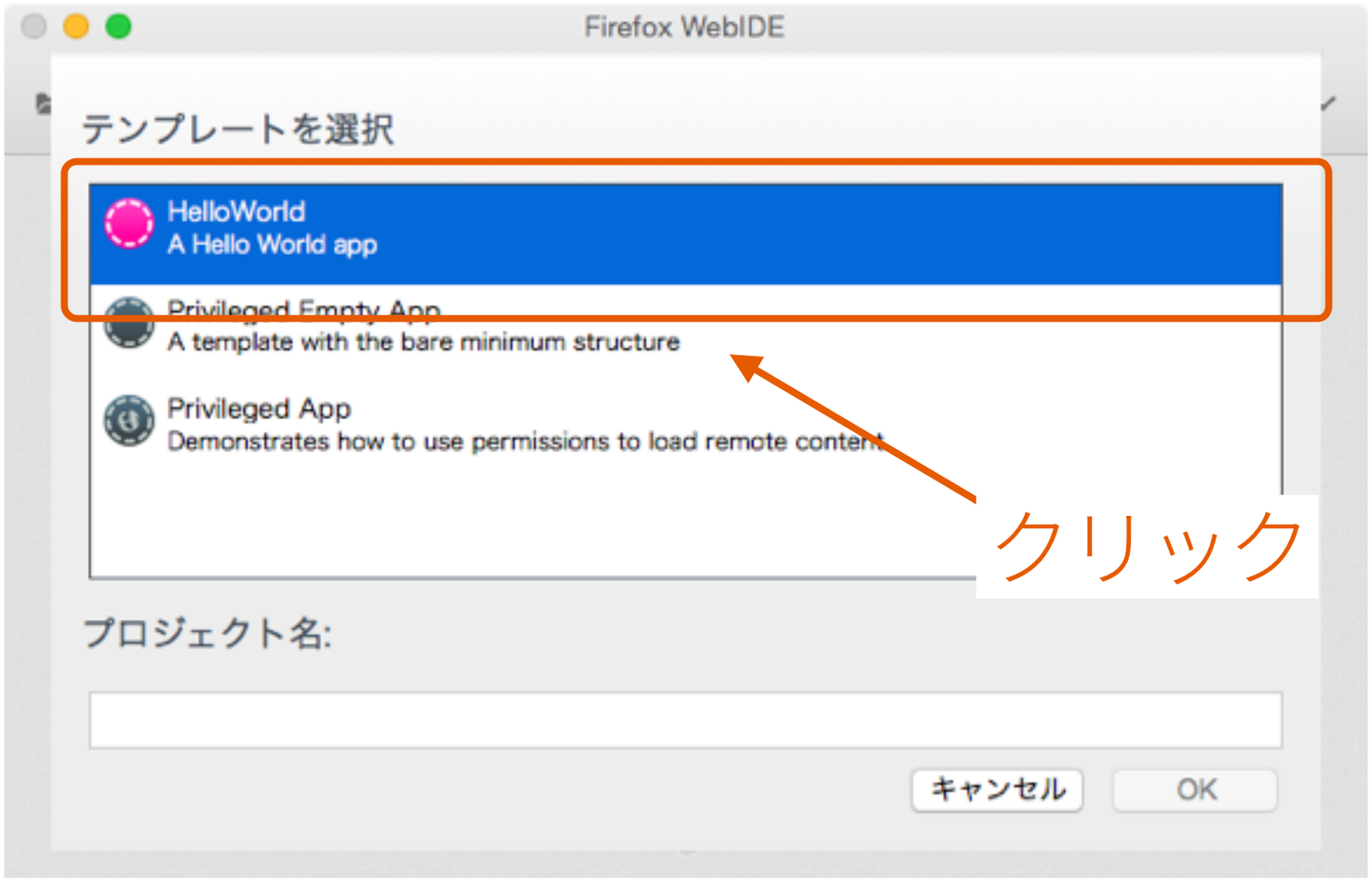
HelloWorld

HelloWorldアプリの作成



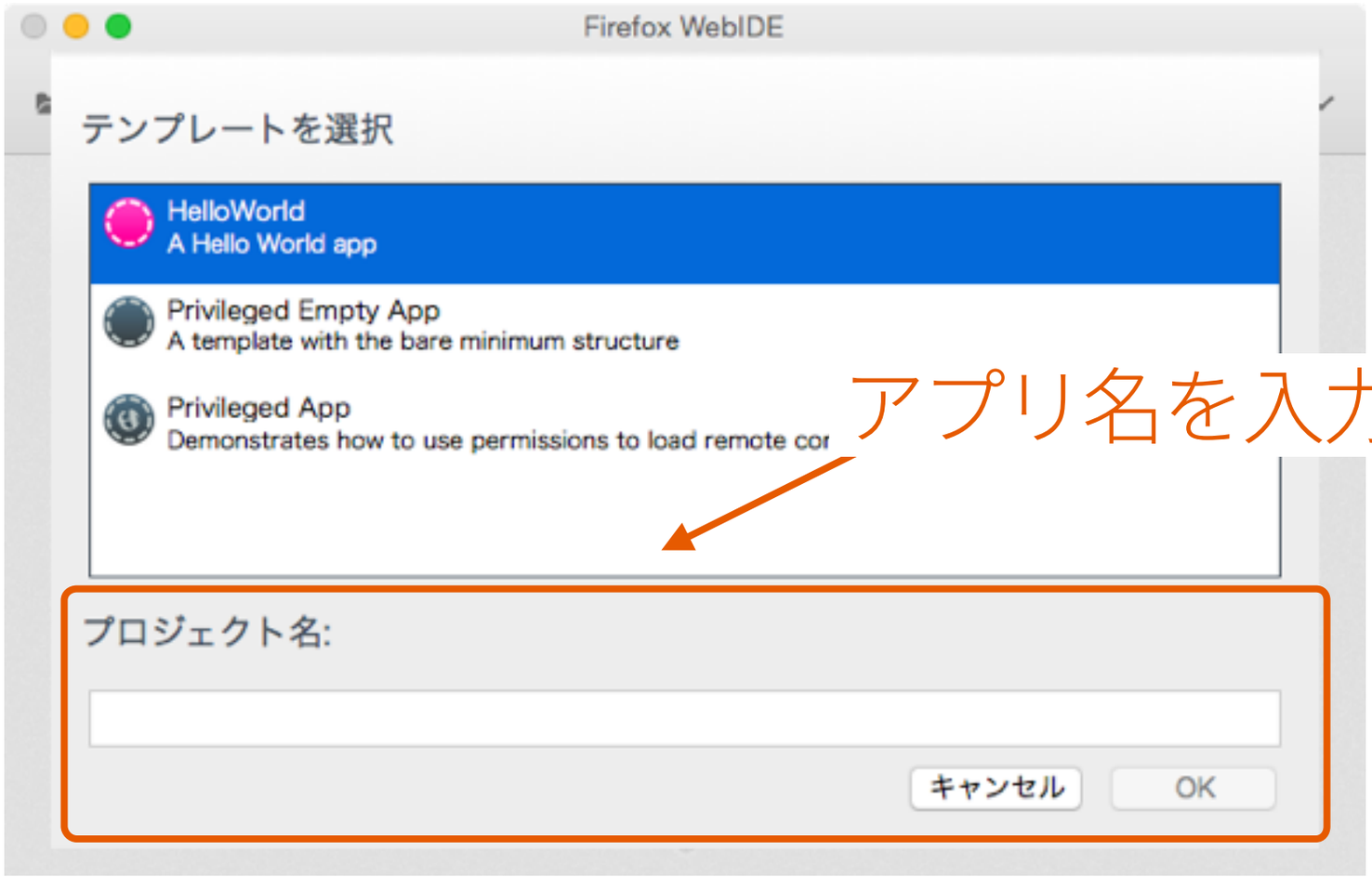
「アプリを開く」から「新規アプリ」を選びます

HelloWorldアプリの作成（つづき）



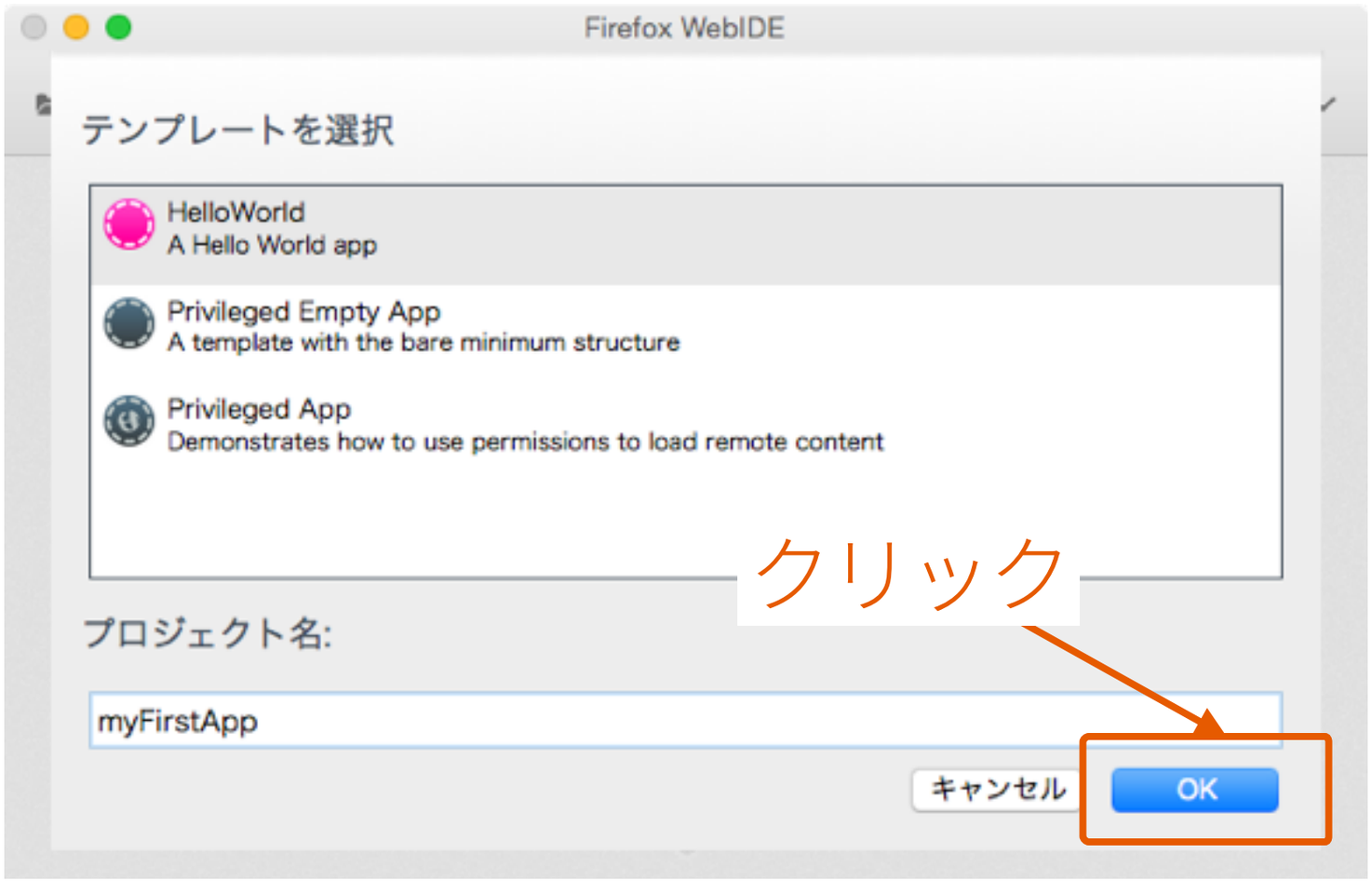
テンプレートの中かから「HelloWorld」を選びます

HelloWorldアプリの作成（つづき）



プロジェクト名にアプリの名前を入力します

HelloWorldアプリの作成（つづき）



OKボタンをクリックし、
テンプレートからアプリを作成します

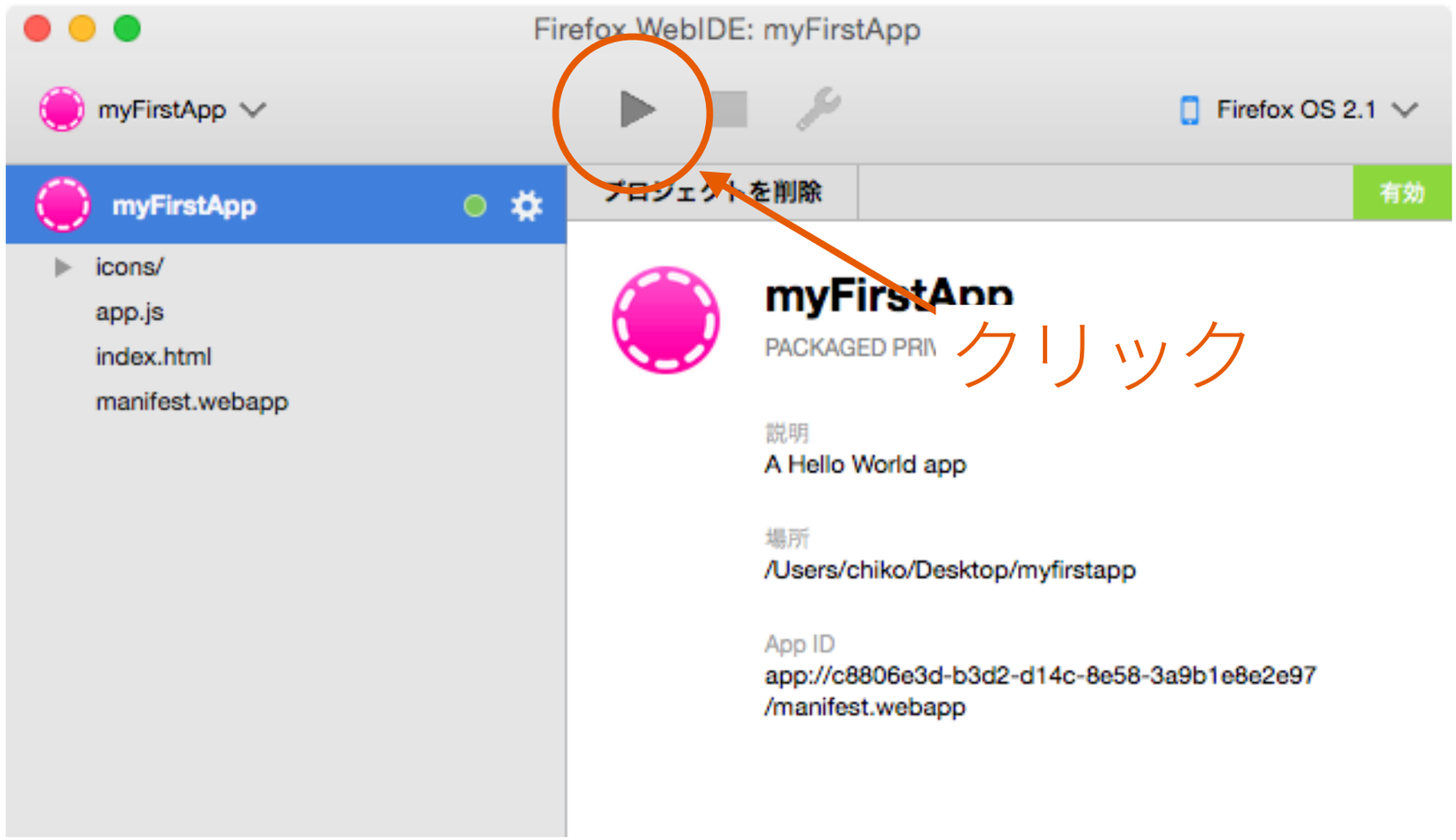
mozilla

HelloWorldアプリの作成（つづき）

このように表示されたアプリ作成終了です

シミュレータで動かそう

HelloWorldアプリの起動



上部中央にある、「▶」ボタンを押すと起動します

起動されたHello Worldアプリ

HelloWorldを改造しよう

改造後のイメージ

改造後のイメージ

文言の変更

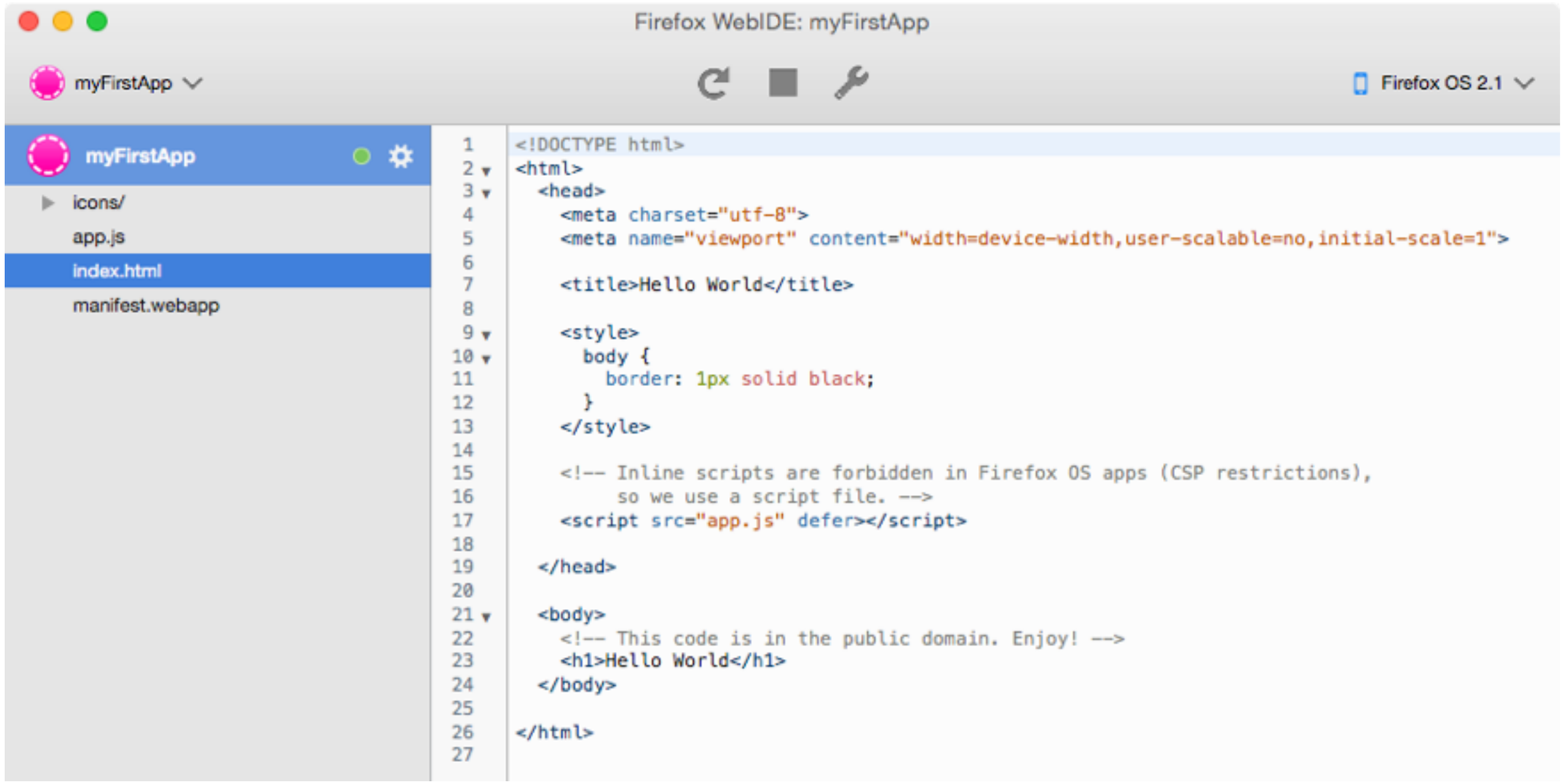
色、文字の大きさの変更

HelloWorldを改造しよう

index.html をクリックすると、右側にソースが表示されます

mozilla

HelloWorldを改造しよう（つづき）



HelloWorldを改造しよう（つづき）

```
<body>
    <h1>こんにちは！せかい！</h1>
</body>
```

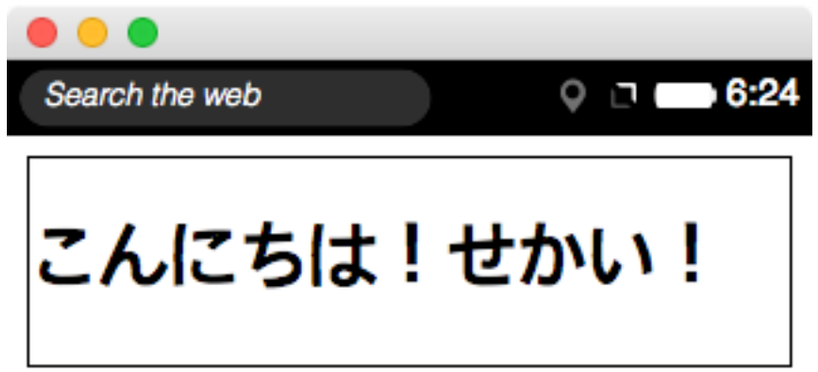
該当部分を上記のように書き換え、保存します

Hello worldを改造しよう（つづき）

再読込ボタンをクリックで、アプリが再起動します。

ショートカットはCtrl-r もしくは Cmd-r です。

Hello worldを改造しよう（つづき）



- 表示される文字が変わりました！
- このように再読み込みを行うことで、変更内容を即座に反映できます。

CSSファイルを追加しよう

head要素を次のように変更します

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,user-
scalable=no,initial-scale=1">

  <title>Hello World</title>
  <link rel="stylesheet" href="app.css">
  <script src="app.js" defer></script>
</head>
```

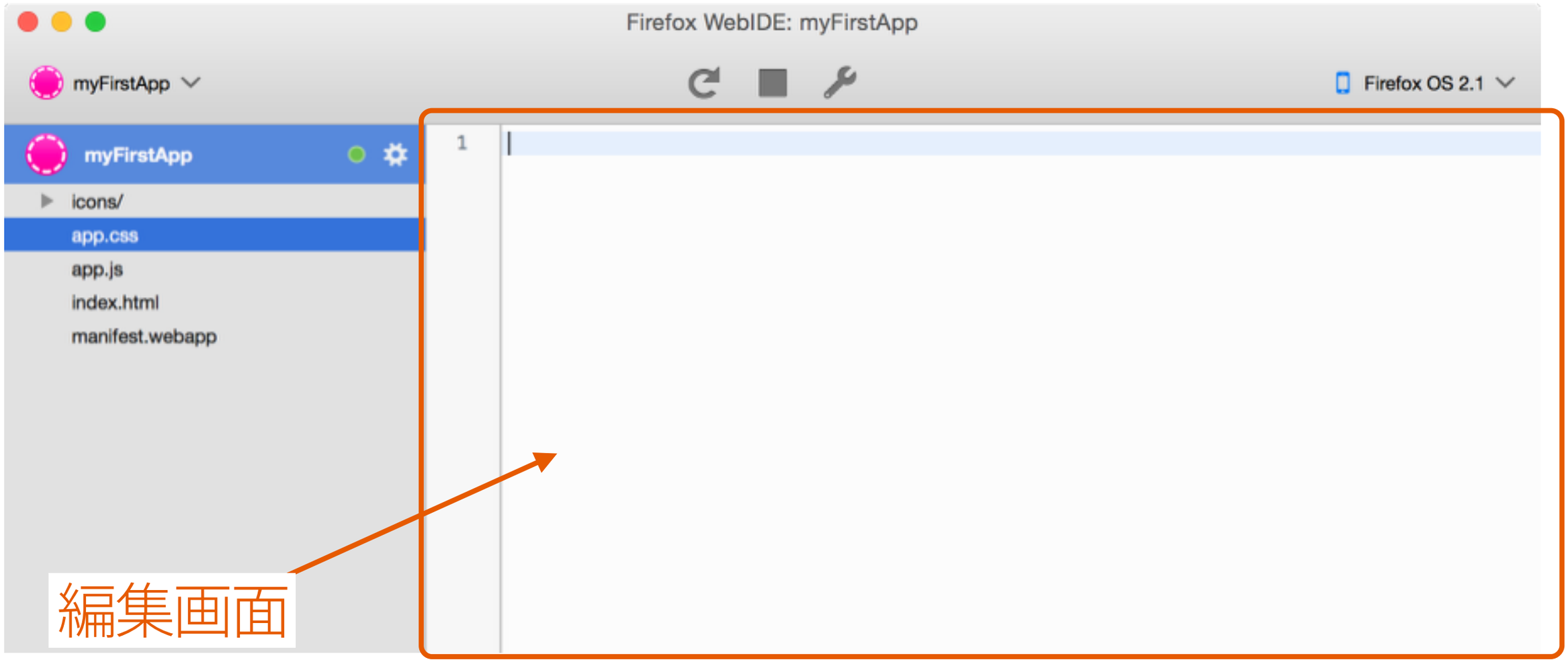
Hello worldを改造しよう（つづき）

画面左のファイルブラウザで右クリック→「新規作成」

Hello worldを改造しよう（つづき）

mozilla

Hello worldを改造しよう（つづき）



空のファイルが作成されます

app.cssの内容

```
body{
  font-size: 18px;
  overflow: hidden;
  background-color: #00539F;
}

h1{
  font-size: 640%;
  line-height: .8;
  color: white;
}
```

センサーの値を利用しよう

環境光センサー

周辺の明るさにあわせて数値が変わる

app.js の変更内容

```
var indicator;

var displayBrightness = function(element, value){
  element.textContent = value;
};

var handleDeviceLightEvent = function(event){
  var lux = event.value;
  if(indicator != null){
    displayBrightness(indicator, lux);
  }
};

window.addEventListener("devicelight", handleDeviceLightEvent);
```

app.js の変更内容（つづき）

```
var initApp = function(){
  indicator = document.querySelector("h1");
};

window.addEventListener("load", initApp);
```

DeviceLight イベント

```
var indicator;

var displayBrightness = function(element, value){
  element.textContent = value;
};

var handleDeviceLightEvent = function(event){
  var lux = event.value;
  if(indicator != null){
    displayBrightness(indicator, lux);
  }
};

window.addEventListener("devicelight", handleDeviceLightEvent);
```

光の強さによって音階が変わるアプリ

周辺が明るくなると音が高くなる

http://goo.gl/PGDLai

モーションセンサー

http://goo.gl/G7XN4k

フォームとの連携

http://goo.gl/QNr7KF

実機で動かそう

実機の設定を開発者向けに変更します

実機の設定変更

タップ

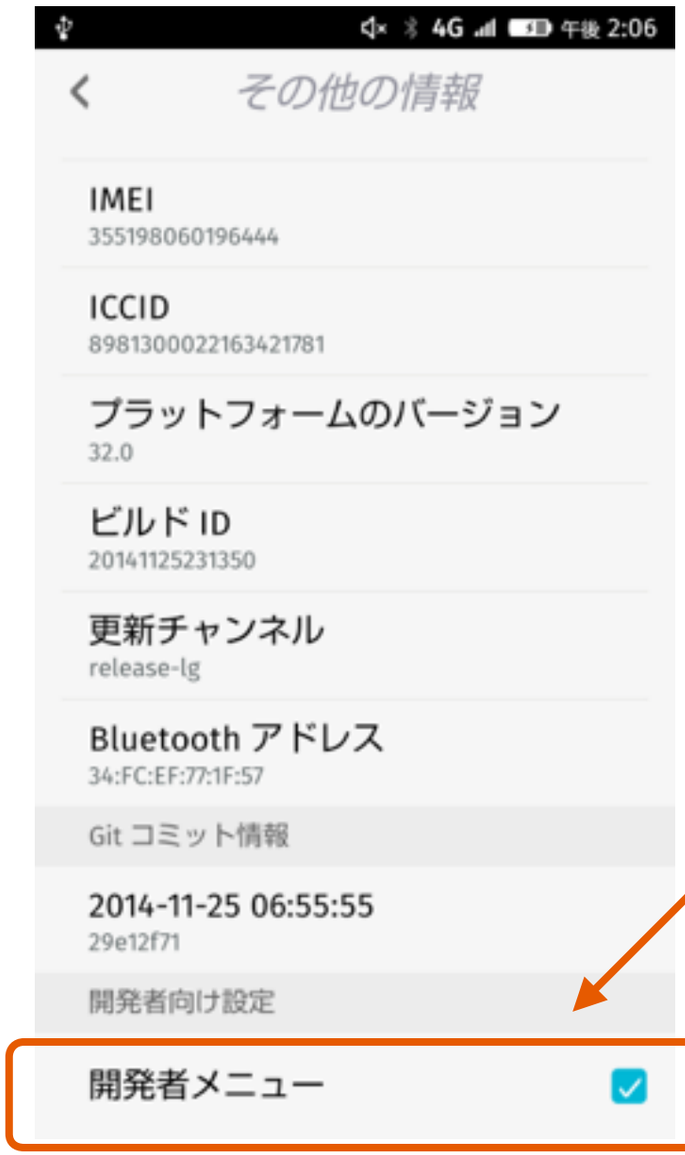
「環境設定アプリ」の端末情報をタップします

実機の設定変更

タップ

「その他の情報」をタップします

実機の設定変更

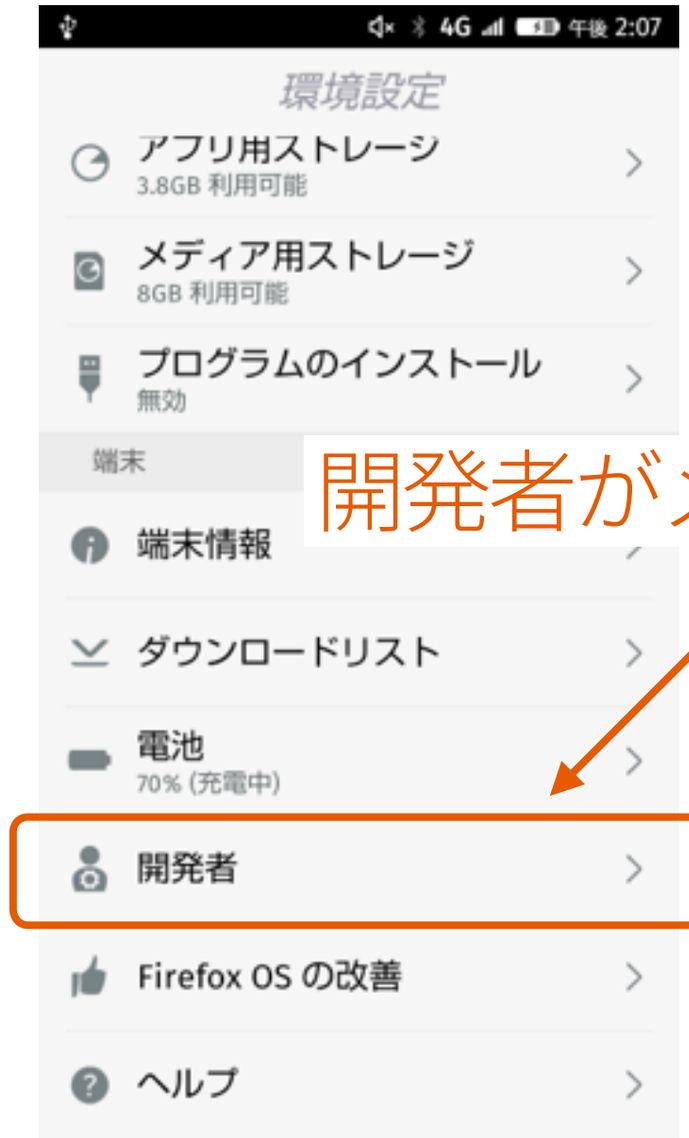


チェック

「開発者メニュー」にチェックを入れます

mozilla

実機の設定変更



アプリのトップに「開発者」が追加されます

実機の設定変更

「開発者」をタップします

実機の設定変更

USB経由のデバッグを「ADBと開発ツール」に変更します

PC側の設定

http://goo.gl/6N7BeP

PCとの接続

WebIDEを利用したPCとの接続

「ランタイムを選択」から端末を選択

シミュレータとの接続と同様に端末を選べば、接続します

カメラを利用しよう

mozilla

カメラからの入力を得るためのAPI

- GetUserMedia
  - ビデオチャット用のAPI
  - ビデオストリームが取得できる
  - 簡単、細かい制御ができない
- CameraAPI
  - カメラアプリを書くためのAPI
  - 撮影、録画、フィルタ、フォーカスの調整、etc
  - 複雑、細かい制御ができる

GetUserMediaを利用する

GetUserMediaを利用した写真撮影の流れ

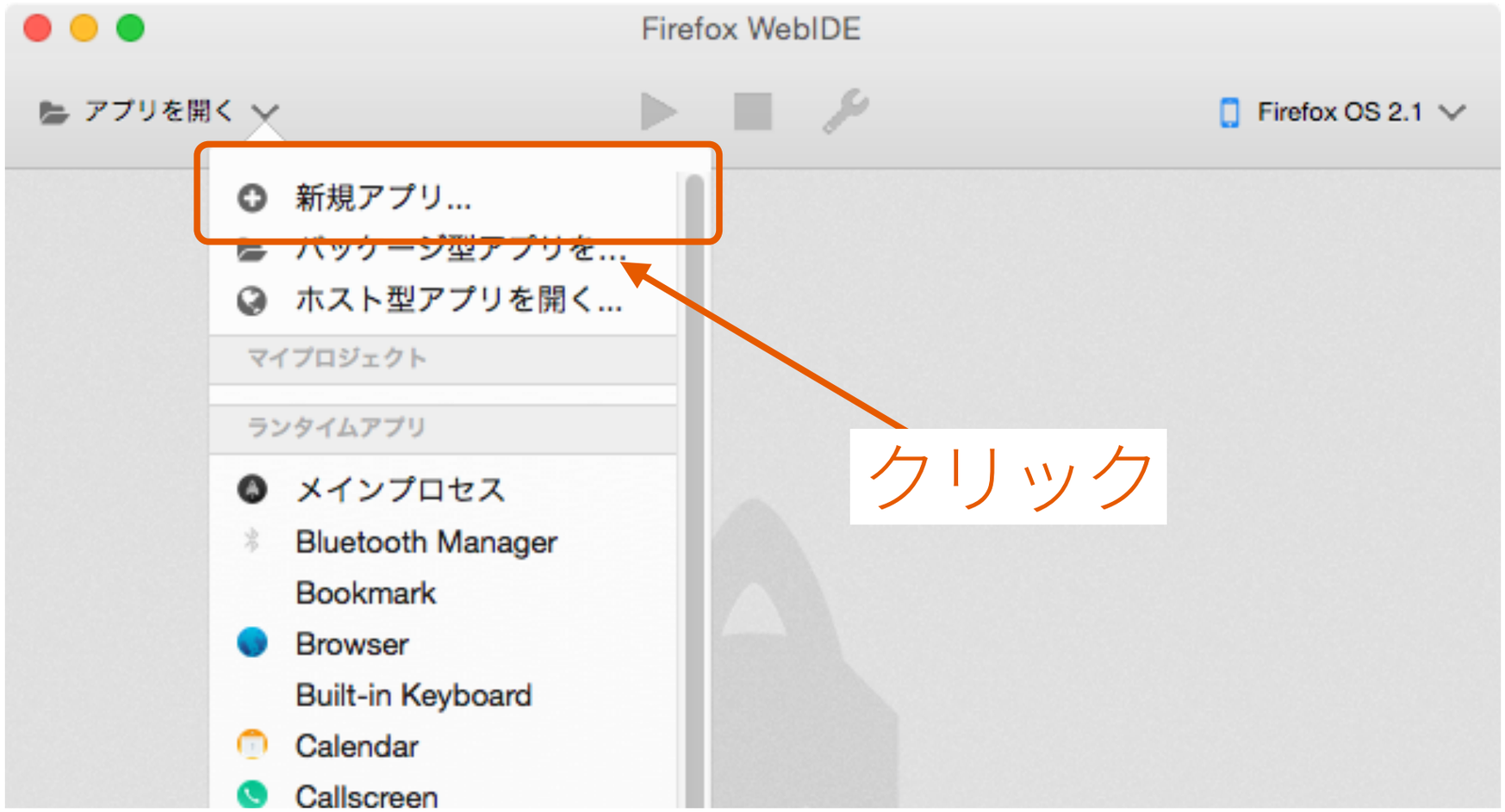
1. カメラからの入力ストリームを得る
2. 得たストリームをvideo要素の入力に指定する
3. Canvasから2Dの描画コンテキストを得る
4. 撮影するタイミングで、
   2のvideo要素を、
   3で得た描画コンテキストに対して描画
5. 4で得た結果をDataURIとして取得

今回作成するアプリ

- 機能
  - GetUserMediaを利用して写真を撮影する
  - 写真をスマホ内の記憶領域に保存する
  - 保存結果OSの通知機能を利用して通知する
- 利用するAPI
  - GetUserMedia
  - DeviceStorage
  - Desktop Notification

完成版：http://goo.gl/9fHYwC

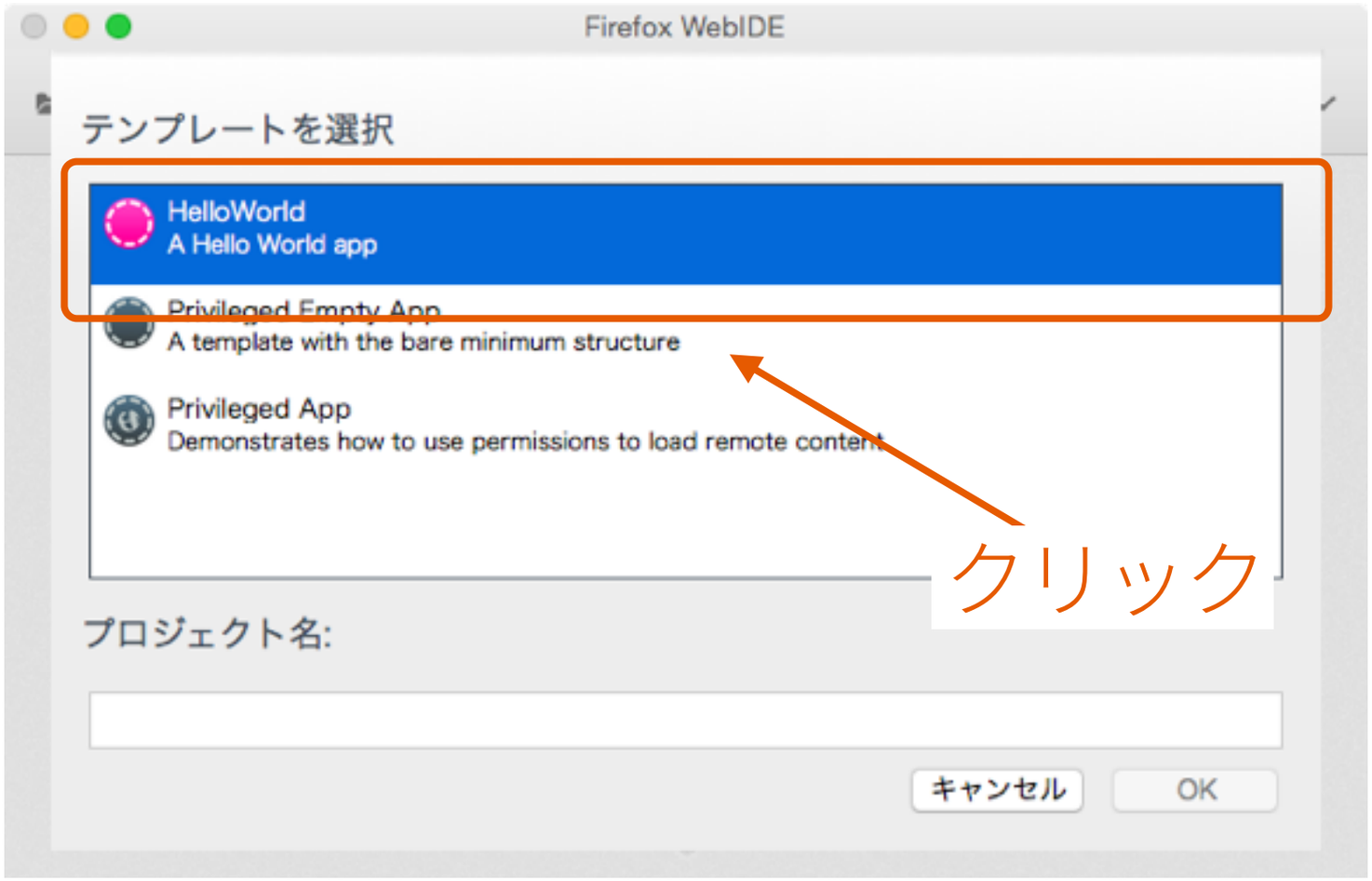
カメラアプリのテンプレート作成



「アプリを開く」から「新規アプリ」を選びます

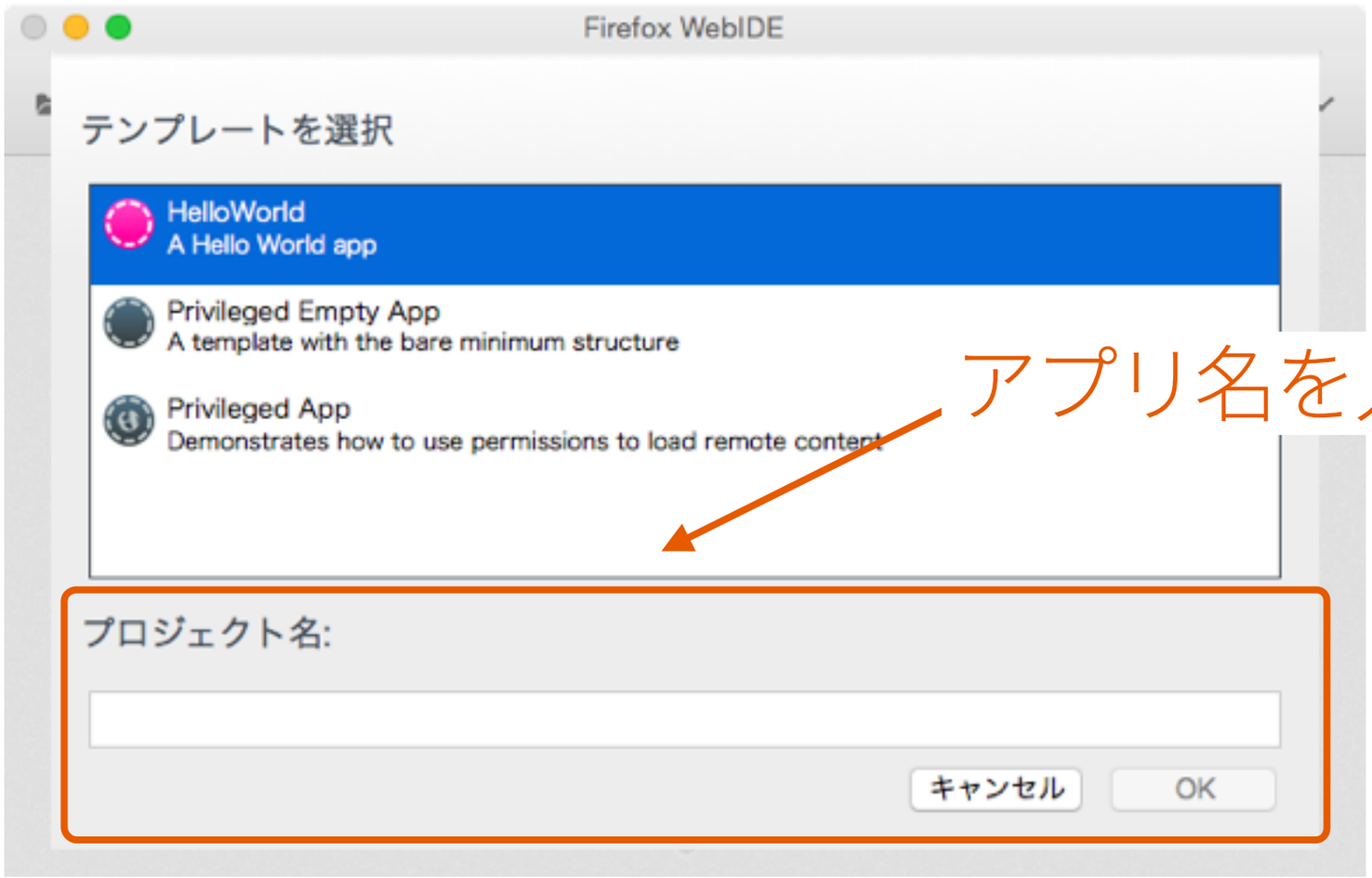
mozilla

カメラアプリのテンプレート作成（つづき）



テンプレートの中かから「HelloWorld」を選びます

カメラアプリのテンプレート作成（つづき）



プロジェクト名にアプリの名前を入力します

manifest.webapp の編集

```
{
  "name": "camera-with-getusermedia",
  "description": "A Hello World app",
  "launch_path": "/index.html",
  "icons": {
    "16": "/icons/icon16x16.png",
    "48": "/icons/icon48x48.png",
    "60": "/icons/icon60x60.png",
    "128": "/icons/icon128x128.png"
  },
  "type": "privileged",
  "permissions": {
    "video-capture": {
      "description": "写真を撮るため"
    },
    "device-storage:pictures": {
      "description": "撮影した写真の保存のため",
      "access": "readwrite"
    },
    "desktop-notification": {
      "description": "写真が保存できたかどうかを通知するため"
    }
  }
}
```

manifest.webapp の編集

```
{
  "name": "camera-with-getusermedia",
  "description": "A Hello World app",
  "launch_path": "/index.html",
  "icons": {
    "16": "/icons/icon16x16.png",
    "48": "/icons/icon48x48.png",
    "60": "/icons/icon60x60.png",
    "128": "/icons/icon128x128.png"
  },
  "type": "privileged",
  "permissions": {
    "video-capture": {
      "description": "写真を撮るため"
    },
    "device-storage:pictures": {
      "description": "撮影した写真の保存のため",
      "access": "readwrite"
    },
    "desktop-notification": {
      "description": "写真が保存できたかどうかを通知するため"
    }
  }
}
```

この部分を追記

manifest.webapp の編集

```
{
  "name": "camera-with-getusermedia",
  "description": "A Hello World app",
  "launch_path": "/index.html",
  "icons": {
    "16": "/icons/icon16x16.png",
    "48": "/icons/icon48x48.png",
    "60": "/icons/icon60x60.png",
    "128": "/icons/icon128x128.png"
  },
  "type": "privileged",
  "permissions": {
   "video-capture": {
     "description": "写真を撮るため"
   },
   "device-storage:pictures": {
     "description": "撮影した写真の保存のため",
     "access": "readwrite"
   },
   "desktop-notification": {
     "description": "写真が保存できたかどうかを通知するため"
   }
  }
}
```

慎重に扱うべきAPIに関しては、
利用を明示する必要があります

manifest.webapp の編集

```
{
  "name": "camera-with-getusermedia",
  "description": "A Hello World app",
  "launch_path": "/index.html",
  "icons": {
    "16": "/i
    "48": "/i
    "60": "/i
    "128": "/
  },
  "type": "pr
  'permissio
    "video-ca
      "description": "写真を撮るため"
    },
    "device-storage:pictures": {
      "description": "撮影した写真の保存のため",
      "access": "readwrite"
    },
    "desktop-notification": {
      "description": "写真が保存できたかどうかを通知するため"
    }
  }
}
```

くわしくは
https://developer.mozilla.org/ja/Apps/App_permissions
をご覧ください

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,user-scalable=no,initial-scale=1">

    <title>Camera With GetUserMedia</title>

    <script src="app.js" defer></script>
    <link rel="stylesheet" href="app.css">
  </head>

  <body>
    <div class="container">
      <video id="preview"></video>
      <canvas id="canvas"></canvas>
      <img id="photo">
    </div>
    <button id="shoot">撮影</button>
  </body>

</html>
```

index.html (body 要素のみ抜粋)

```
<body>
  <div class="container">
    <video id="preview"></video>
    <canvas id="canvas"></canvas>
    <img id="photo">
  </div>
  <button id="shoot">撮影</button>
</body>
```

app.css

```
body{
  margin: 0;
  padding: 0;
}
.container{
  padding-bottom: 4em;
  width: 100%;
}
#preview{
  width: 100%;
}
#canvas{
  display: none;
}
#shoot{
  font-size: 100%;
  width: 100%;
  padding: 1em;
  position: fixed;
  bottom: 0;
  left: 0;
}
#photo{
  width:100%;
}
```

app.js：変数と定数

```
navigator.getUserMedia =
  navigator.getUserMedia || navigator.mozGetUserMedia;

const SIZE = {
  width: 800,
  height: 600
};

var storage;
var elements;
var gc;
```

app.js：アプリの初期化

```
var initApp = function(){
  elements = {
    preview: document.querySelector("#preview"),
    canvas: document.querySelector("#canvas"),
    photo: document.querySelector("#photo"),
    shoot: document.querySelector("#shoot")
  };
  gc = elements.canvas.getContext("2d");
  // キャンバスの解像度をサイズに合わせる
  elements.canvas.width = SIZE.width;
  elements.canvas.height = SIZE.height;
  getVideoStream().then(setPreviewStream, log);
  elements.preview.addEventListener("canplay", enableShootButton);

  storage = navigator.getDeviceStorage("pictures");
};
window.addEventListener("load", initApp);
```

app.js：カメラからの入力ストリーム獲得

```
var doGetVideoStream = function(resolve, reject){
  navigator.getUserMedia({video: true, audio: false},
    function(stream){
      window.addEventListener("unload", function(event){
        stream.stop(); // アプリ終了時にストリームを停止
      });
      resolve(stream);
  }, reject);
};

var getVideoStream = function(){
  return new Promise(doGetVideoStream);
};
```

app.js：カメラからの入力の表示

```
var setPreviewStream = function(stream){
  if(elements != null && elements.preview != null){
    elements.preview.src =
      window.URL.createObjectURL(stream);
    elements.preview.play();
  }
};
```

app.js：撮影ボタンの有効化

```
var enableShootButton = function(){
  if(elements != null && elements.shoot){
    elements.shoot.addEventListener("click", takePhoto);
  }
};
```

app.js：撮影機能

```
var takePhoto = function(){
  if(gc != null && elements != null &&
     elements.preview != null && elements.canvas != null){

    gc.drawImage(elements.preview,
                 0, 0,
                 SIZE.width, SIZE.height);
    var blob = elements.canvas.toBlob(showAndSavePhoto,
                                      "image/jpg");
  }
};
```

app.js：撮影した写真の表示と保存

```
var createFilename = function(){
  return "handson-20130310-"+ (new Date()).valueOf() + ".jpg";
};
var showAndSavePhoto = function(blob){
  if(elements != null && elements.photo != null){
    elements.photo.setAttribute("src",
                                window.URL.createObjectURL(blob));
  }
  if(storage != null){
    var request = storage.addNamed(blob, createFilename());
    request.onsuccess = showSavingPhotoSuccess;
    request.onerrorr = showSavingPhotoError;
  }
};
```

app.js：通知

```
var sendNotification = function(message){
  new Notification(message);
};
var showSavingPhotoSuccess = function(){
  sendNotification("写真が保存されました");
};
var showSavingPhotoError = function(){
  var msg = "写真の保存に失敗しました";
  log(msg);
  sendNotification(msg);
};
var log = function(msg){
  console.log(msg);
};
```

まとめ

すべてWeb技術でできています！

通知、プッシュ通知 →

電話、SMS →

カメラ →

ホーム画面、壁紙 →

SDカード →

メール →

← ネットワーク、電池

← ブラウザ

← 動画、音楽

← 通信料管理

mozilla

詳細はMDNのアプリセンターへ！

https://developer.mozilla.org/Apps