報道各位

2019年6月7日
株式会社インプレスR&D
https://nextpublishing.jp/

ブラウザで動くカメラアプリの作り方をハンズオン形式で学ぶ！

『カメラアプリで体感するWeb App』発行

技術の泉シリーズ、6月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレスR&Dは、『カメラアプリで体感するWeb App』（著者：宮代 理弘）を発行いたします。

最新の知見を発信する『技術の泉シリーズ』は、「技術書典」をはじめとした各種即売会や、勉強会・LT会などで頒布された技術同人誌を底本とした商業書籍を刊行し、技術同人誌の普及と発展に貢献することを目指します。

『カメラアプリで体感するWeb App』
https://nextpublishing.jp/isbn/9784844398950

著者：宮代 理弘
小売希望価格：電子書籍版 2200円（税別）／印刷書籍版 2400円（税別）
電子書籍版フォーマット：EPUB3／Kindle Format8
印刷書籍版仕様：B5判／カラー／本文196ページ
ISBN：978-4-8443-9895-0
発行：インプレスR&D

＜＜発行主旨・内容紹介＞＞

本書は、「今」使える技術やAPIをふんだんに使ったブラウザで動くカメラアプリの作り方を通じて、各種の技術をハンズオン形式で解説します。

Parcel や React、CSS Modules の設定に始まり、画面のスタイリング、 getUserMedia や applyConstraints、ImageCapture、 EXIF、 Geolocation、 WebAssembly、Web Worker、OffscreenCanvas、Shape detectipn API、WebGL、PWA などの技術を、『広く浅く』学ぶことができる一冊です！

（本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。）

一つ一つの機能を加えながら、カメラアプリを完成させていきます。

第3章　カメラの設定を変えよう

3.1　カメラの最大解像度を調べる

最低限のカメラアプリはできましたが、まだまだ足りない機能がたくさんあります。まずは解像度の設定を行いましょう。前章のコードで、getUserMedia()に何を渡していたでしょうか。

```
navigator.mediaDevices.getUserMedia({
  video: {
    aspectRatio: { ideal: 16 / 9 },
    facingMode: { ideal: 'environment' },
  },
});
```

このaspectRatioで、アスペクト比を設定すると説明しました。では、解像度についてはどうやって設定するのでしょうか。解像度は、widthとheightで設定できます。また、ここまで挙げた設定では、min/max/ideal/exactの指定ができます[1]。exactで指定すると、必ずその値になるようなカメラデバイスを選択します。条件を満たせないときには、OverconstrainedErrorが返されます。idealで指定すると、その値に極力合わせたカメラデバイスを選択します。条件を満たせないときには、minとmaxの間でidealに近い値が利用されます。

```
navigator.mediaDevices.getUserMedia({
  video: {
    width: { min: 360, max: 1080, ideal: 720 },
    height: { min: 460, max: 1920, ideal: 1280 },
  },
});
```

最大解像度を取得するコードを書いてみましょう。最大解像度を得るAPIはないため、よく使われる解像度を総当たりで試して探します。getConstraints()は、特定のfacingModeにおける推奨設定を作る関数です。調べる解像度は、RESOLUTION_LISTに定めています。

isResolutionAvailable()は、解像度が利用できるかを調べる関数です。getUserMedia()は、指定した条件を満たすものがない場合はエラーを返します。エラーの内容は今回必要としていないため、catchしてnullを返すようにします。これで、nullかどうかで成功したかがわかります。

デバイスによっては、指定した条件で返ってきたストリームの解像度が違う場合もあります。そのため、一旦<video>を作ってから、実際の解像度を調べる必要があります。条件の設定では長辺をwidthとしていますが、取得される動画は必ずしも**横幅が長辺とは限りません**。比較するためには、得られた動画の長辺を調べる必要があります。さいごに、使い終わった<video>やトラックは消しておきましょう。

src/helpers/getConstraints.js

```
import createVideoElement from '~/helpers/createVideoElement';

const RESOLUTION_LIST = [
  { width: 7680, height: 4320 }, // 8K
  { width: 3840, height: 2160 }, // 4K
  { width: 2560, height: 1440 }, // WQHD
  { width: 1920, height: 1080 }, // Full-HD
  { width: 1280, height: 720 }, // HD
];

/**
 * @param {'user' | 'environment'} facingMode
 * @returns {Promise<MediaTrackConstraints | null>}
 */
async function getConstraints(facingMode) {
  for (const resolution of RESOLUTION_LIST) {
    if (await isResolutionAvailable(resolution, facingMode)) {
      return {
        width: { exact: resolution.width },
        height: { exact: resolution.height },
        facingMode: { exact: facingMode },
      };
    }
  }

  return null;
}

/** @param {MediaTrackConstraints} resolution */
async function isResolutionAvailable(resolution, facingMode) {
  const stream = await navigator.mediaDevices
    .getUserMedia({
      video: {
        width: { exact: resolution.width },
        height: { exact: resolution.height },
```

1. 詳細については https://developer.mozilla.org/ja/docs/Web/API/MediaDevices/getUserMedia を参照してください

フィルターの実装方法も丁寧に解説します。

CanvasをoffscreenCanvasとして、引数に渡すのが良いでしょう。今回のフィルターは次のような関数として定義されます。

```
/**
 * @param {HTMLCanvasElement | OffscreenCanvas} canvas
 * @param {ImageBitmap} bitmap
 */
async function fitlerFunction (canvas, bitmap) {}
```

しかし、実際の元データはBlobになるため、BlobからImageBitmapを作成させてからフィルターに通す必要があります。applyFilter()では、createImageBitmapでBlobを変換します。フィルター後の画像を出力するための<canvas>には、読み込んだ画像と同じサイズを設定します。このあと、~/filtersにはフィルターをいくつか定義していきます。フィルター関数はfilters[filterType]として取得でき、関数がない場合には同じ画像を返すようにします。

src/helpers/applyFilter.js

```
import * as filters from '~/filters';

/**
 * @param {Blob} blob
 * @param {string} filterType
 * @returns {Promise<ImageBitmap>}
 */
async function applyFilter(blob, filterType) {
  const bitmap = await createImageBitmap(blob);

  const canvas = document.createElement('canvas');
  Object.assign(canvas, {
    width: bitmap.width,
    height: bitmap.height,
  });

  const filterFn = filters[filterType];
  if (!filterFn) {
    return bitmap;
  } else {
    await filters[filterType](canvas, bitmap);
  }

  const result = await createImageBitmap(canvas);
  return result;
}

export default applyFilter;
```

6.3　JavaScriptでフィルターを実装する

まずは簡単なグレースケールフィルターを作ってみましょう。

ImageBitmapは単体では何もできませんが、CanvasImageSourceになるのでctx.drawImage()に渡すことができます。そして、ctx.getImageData()によってImageDataに変換できます。

ImageDataはImageData.dataにUint8ClampedArrayのバッファーがあり、バッファーから画像を加工することができます。このバッファーには、RGBとアルファ値の4つが連続して並んでいます。つまり、ピクセル単位で加工する際には4つずつデータを取得する必要があります。

加工し終わったImageDataをCanvasに渡したいところですが、CanvasImageSourceではないためctx.drawImage()に渡すことはできません。そこで、代わりにctx.putImageData()によってCanvasに貼り付けます。

グレースケールの計算ですが、単純に足し合わせて3で割る方法ではうまくいきません（図6.3）。

一般的に青は暗く見え、緑は明るく見えます。そのため、RGBそれぞれの値について補正する必要があります。今回は、テレビ放送やJPEGなどで利用される"ITU-R BT.601"[4]に則って、輝度値Yを計算します。この輝度値をRGBに割り当ててグレースケール化します。

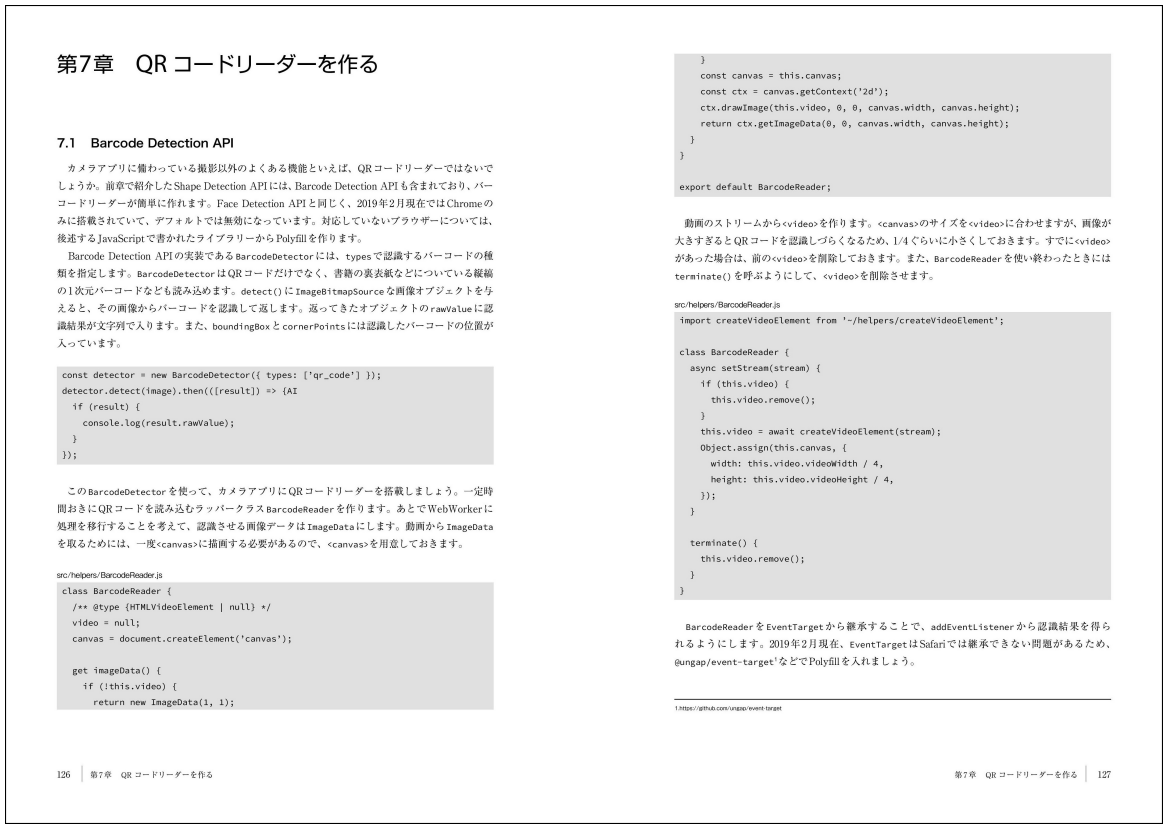
図6.3: 平均値とITU-R BT.601の違い

src/filters/purejs/grayscale.js

```
/**
 * @param {HTMLCanvasElement} canvas
 * @param {ImageBitmap} bitmap
 */
function grayscale(canvas, bitmap) {
  const ctx = canvas.getContext('2d');
```

4. 以前はCCIR 601と呼ばれていました

カメラの用途としてポピュラーな QR コードリーダーも実装します。



第7章　QR コードリーダーを作る

7.1　Barcode Detection API

カメラアプリに備わっている撮影以外のよくある機能といえば、QR コードリーダーではないでしょうか。前章で紹介した Shape Detection API には、Barcode Detection API も含まれており、バーコードリーダーが簡単に作れます。Face Detection API と同じく、2019 年 2 月現在では Chrome のみに搭載されていて、デフォルトでは無効になっています。対応していないブラウザーについては、後述する JavaScript で書かれたライブラリーから Polyfill を作ります。

Barcode Detection API の実装である BarcodeDetector には、types で認識するバーコードの種類を指定します。BarcodeDetector は QR コードだけでなく、書籍の裏表紙などについている縦縞の 1 次元バーコードなども読み込めます。detect() に ImageBitmapSource な画像オブジェクトを与えると、その画像からバーコードを認識して返します。返ってきたオブジェクトの rawValue に認識結果が文字列で入ります。また、boundingBox と cornerPoints には認識したバーコードの位置が入っています。

```
const detector = new BarcodeDetector({ types: ['qr_code'] });
detector.detect(image).then(([result]) => {AI
  if (result) {
    console.log(result.rawValue);
  }
});
```

この BarcodeDetector を使って、カメラアプリに QR コードリーダーを搭載しましょう。一定時間おきに QR コードを読み込むラッパークラス BarcodeReader を作ります。あとで WebWorker に処理を移行することを考えて、認識させる画像データは ImageData にします。動画から ImageData を取るためには、一度<canvas>に描画する必要があるので、<canvas>を用意しておきます。

src/helpers/BarcodeReader.js

```
class BarcodeReader {
  /** @type {HTMLVideoElement | null} */
  video = null;
  canvas = document.createElement('canvas');

  get imageData() {
    if (!this.video) {
      return new ImageData(1, 1);
    }
    const canvas = this.canvas;
    const ctx = canvas.getContext('2d');
    ctx.drawImage(this.video, 0, 0, canvas.width, canvas.height);
    return ctx.getImageData(0, 0, canvas.width, canvas.height);
  }
}

export default BarcodeReader;
```

動画のストリームから<video>を作ります。<canvas>のサイズを<video>に合わせますが、画像が大きすぎると QR コードを認識しづらくなるため、1/4 ぐらいに小さくしておきます。すでに<video>があった場合は、前の<video>を削除しておきます。また、BarcodeReader を使い終わったときには terminate() を呼ぶようにして、<video>を削除させます。

src/helpers/BarcodeReader.js

```
import createVideoElement from '~/helpers/createVideoElement';

class BarcodeReader {
  async setStream(stream) {
    if (this.video) {
      this.video.remove();
    }
    this.video = await createVideoElement(stream);
    Object.assign(this.canvas, {
      width: this.video.videoWidth / 4,
      height: this.video.videoHeight / 4,
    });
  }

  terminate() {
    this.video.remove();
  }
}
```

BarcodeReader を EventTarget から継承することで、addEventListener から認識結果を得られるようにします。2019 年 2 月現在、EventTarget は Safari では継承できない問題があるため、@ungap/event-target[1] などで Polyfill を入れましょう。

1 https://github.com/ungap/event-target

126 | 第7章　QR コードリーダーを作る

第7章　QR コードリーダーを作る | 127

＜＜目次＞＞

＜＜著者紹介＞＞

宮代 理弘
Web フロントエンドデベロッパー。「ひとは本質に注力し、すべきことのみするべき」を信念にプロダクトをつくっている。学生クリエイターとしてクマ財団 1 期生に選抜され、KUMA EXHIBITION 2018 では、文章を縦書きで読むためのウェブアプリ『タテ読ミ』を展示した。また、技術系同人サークル『O'CREILLY』では主宰を務め、コミックマーケットや技術書典などで出展している。
Web サイト：https://3846masa.dev
Twitter / GitHub：@3846masa

＜＜販売ストア＞＞

電子書籍：
Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER
印刷書籍：
Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス

※　各ストアでの販売は準備が整いしだい開始されます。
※　全国の一般書店からもご注文いただけます。

【インプレス R&D】 https://nextpublishing.jp/
株式会社インプレスR&D（本社：東京都千代田区、代表取締役社長：井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishing を使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。
※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム（またはメソッド）の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド（POD）による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/
株式会社インプレスホールディングス（本社：東京都千代田区、代表取締役：唐島夏生、証券コード：東証 1 部 9479）を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「旅・鉄道」「学術・理工学」を主要テーマに専門性の高いメデイア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】
株式会社インプレス R&D NextPublishing センター
TEL 03-6837-4820
電子メール: np-info@impress.co.jp