



第 3 回

スマホで 様々なメディアを扱う

ここからしばらくはスマホがメインです。

タッチパネルやカメラなど、スマホ特有のデバイスを
HTMLとJavaScriptで使い倒します。

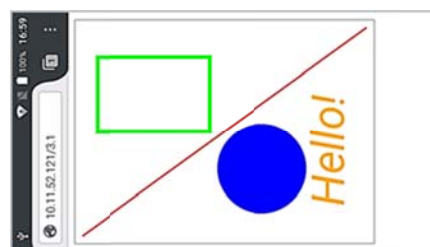
✧ 3.0 スマホで様々なメディアを扱う

ここからは「スマホならではの」機能を取り扱っていきます。今回は、タッチを用いたお絵描き、スマホの標準カメラアプリやカメラロールへのアクセス、画像ファイルの保存、動画や音声の再生、標準カメラアプリを用いないオリジナルカメラアプリ、音声合成アプリを作ります。スマホらしい様々なメディアを HTML と JavaScript で扱うための基本的な方法を学びましょう。

3.1 Canvas に描画

→ 3 ページ

Canvas は、グラフィックス、アニメーション、ゲームなどに不可欠な、画用紙やお絵描きセットのような部品です。Canvas 上に、線、四角、円、文字を描きながら、Web アプリにおける描画の基本を学びます。



3.2 タッチでお絵かき

→ 7 ページ

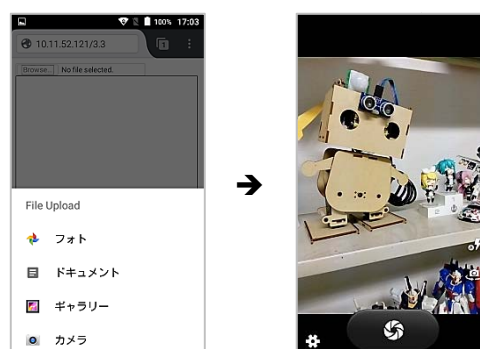
スマホといえばタッチパネルが特徴です。タッチの情報を取得して、Canvas 上に指でお絵かきできるアプリを作りながら、タッチイベントの使い方を学びます。



3.3 写真を撮る・見る

→ 11 ページ

スマホの標準カメラアプリで写真を撮ったり、カメラロールの写真を選んで表示します。ローカル（スマホ内）にあるファイルを Web アプリで扱う方法を紹介します。



3.4 写真に落書き

→ 15 ページ

3.1～3.3のテクニックを組み合わせ、撮った写真やカメラロールにある写真にタッチで落書きして、さらにそれをローカル（スマホ内）に保存できるアプリを作ります。



3.5 動画と音声

→ 19 ページ

つい最近まで Web アプリで動画や音声を扱うには Flash を使う必要がありました。今は、HTML だけで動画や音声を扱えます。超簡単なメディアプレイヤーアプリを作ります。



3.6 My カメラアプリ

→ 21 ページ

標準カメラアプリを使わずに、カメラデバイスに直接アクセスして映像を取り込む方法を紹介합니다。カメラを用いたオリジナルアプリを作る基礎を学びます。



3.7 音声合成アプリ

→ 25 ページ

音声は自然で直感的なインタラクションに欠かせません。ブラウザの音声合成機能を使い、自由に入力したテキストを話すアプリを作ります。また、話した言葉をテキストに変換する音声認識の方法も紹介します。



本日の課題

→ 28 ページ

✈ 3.1 Canvas に描画

	PC	And	iOS
Firefox	○	○	○
Chrome	○	○	○

Canvas は、グラフィックス、アニメーション、ゲームなどに不可欠な、画用紙やお絵描きセットのような部品です。Canvas 上に、線、四角、円、文字を描きながら、Web アプリにおける描画の基本を学びます。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 3.1 の中に、1.1 で作った index.html (テンプレート) をコピーします。
- この index.html を Brackets で開き、以下のコードを入力しましょう。長めですが頑張りましょう。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 3.1 ¥ index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <meta name='viewport' content='width=device-width,initial-scale=1'>
    <title>3.1 Canvas に描画</title>
    <style>
      #canvas {
        /* Canvas に枠をつける CSS */
        border: solid 1px #000000;
      }
    </style>
  </head>
  <body>
    <!-- Canvas (お絵描きセット) を置く -->
    <canvas width='300' height='400' id='canvas'></canvas> ①
    <script>
      // Canvas を使うためのお決まりの 2 行 ②
      const canvas = document.getElementById('canvas'); // お絵描きセットを得て
      const context = canvas.getContext('2d'); // その筆を得る
      // Canvas に線を描く
      context.beginPath(); // 描き始める ③
      context.moveTo(10, 10); // 筆の初期位置 ④
      context.lineTo(290, 390); // 筆を動かす ⑤
      context.strokeStyle = '#FF0000'; // 筆の色 ⑥
      context.lineWidth = 2; // 筆の太さ ⑦
      context.stroke(); // 線を描く ⑧
      // Canvas に四角を描く (線だけ)
      context.beginPath(); // 描き始める
      context.strokeStyle = '#00FF00'; // 筆の色
      context.lineWidth = 5; // 線の太さ
      context.strokeRect(150, 30, 100, 150); // 矩形(Left,Top,Width,Height) ⑨
```

```
// Canvas に円（弧）を描く（塗りつぶし）
context.beginPath(); // 描き始める
context.arc(100, 250, 60, 0, 2*Math.PI); // 円弧(X,Y,半径,開始角,終了角) ⑩
context.fillStyle = '#0000FF'; // 塗りつぶしの色 ⑪
context.fill(); // 塗りつぶす ⑫

// 文字を描く（塗りつぶし）
context.fillStyle = '#FFA500'; // 塗りつぶしの色
context.font = 'italic 60px sans-serif'; // 装飾、サイズ、書体
context.fillText('Hello!', 50, 360); // fillText(text,X,Y) ⑬
</script>
</body>
</html>
```



スマホで動作確認

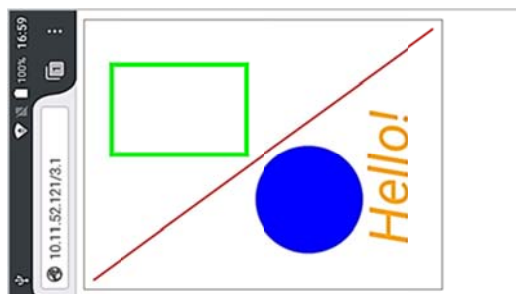
- Web サーバをスタートさせ、アクセス用のアドレスを調べる

- **XAMPP Control Panel** を開き、Apache を **Start**
- **コマンドプロンプト** を開き、**ipconfig** して **IPv4 アドレス** をメモする

- スマホの **Firefox** で確認

- メモしたアドレス（★）に続けて「/3.1/」と入力して Enter
- アドレス例： **10.11.123.45/3.1/**

メモ
★



※ 紙面の都合で横倒しに表示しています



解説

- ① `<canvas width='300' height='400' id='canvas'></canvas>`

絵や画像の入れ物となる `<canvas>` タグ です。スマホ画面に合うように幅と高さを指定しました。

- ② **Canvas を使うためのお決まりの 2 行**

Canvas を使って描画するには始めに必ずこの 2 行を書きます。まずは canvas 要素を取得して、`const context = canvas.getContext('2d');` で `context`（絵筆と思えばよい）を得ます。`getContext()` の () 内には現状 `'2d'` 以外のキーワードは入りません。

③ **context.beginPath();**

Canvas に線を引く場合は必ず beginPath() で始めます。これから描き始める、という意味です。

④ **context.moveTo(10, 10);**

絵筆を初期位置に動かす moveTo() です。() 内には Canvas 内の X 座標 (横方向) と Y 座標 (縦方向) を指定します。Canvas の左上が (0, 0) です。

⑤ **context.lineTo(290, 390);**

線を引く lineTo() です。④の初期位置から () 内の座標に向かって線を引きます。ただし、この段階ではまだ「見えない線」です。

⑥ **context.strokeStyle = '#FF0000';**

線の色を指定する strokeStyle() です。色は 16 進数コードで指定します。

⑦ **context.lineWidth = 2;**

線の幅 (太さ) を指定する lineWidth() です。太さの単位はピクセル (px) です。この命令を省略すると 1 (px) になります。

⑧ **context.stroke();**

この stroke() でようやく、⑥の色、⑦の太さで線が描かれます。「線を引く時は③～⑧をセットで使う」とおぼえておきましょう。

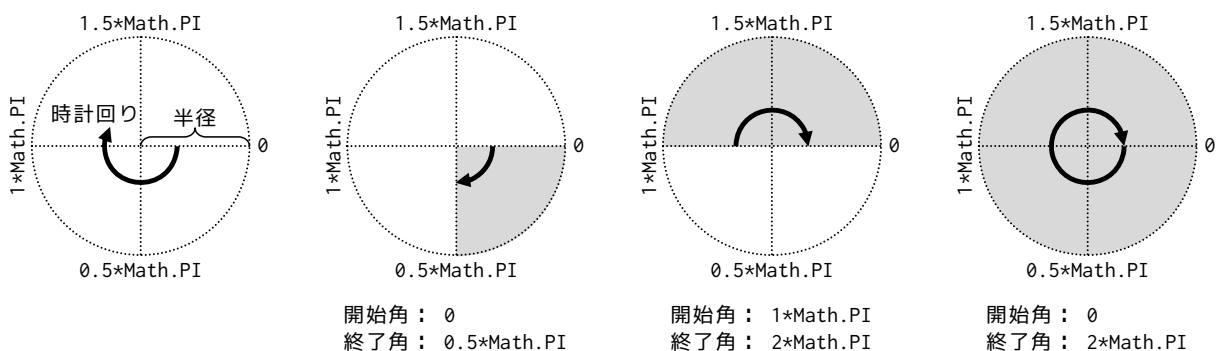
⑨ **context.strokeRect(150, 30, 100, 150);**

四角形 (矩形) を線で描く strokeRect() です。() 内は、左上の X (Left), Y (Top), 幅 (Width), 高さ (Height) を指定します。

塗りつぶし四角形を描く fillRect() というものもあります。

⑩ **context.arc(100, 250, 60, 0, 2*Math.PI);**

円弧を描く arc() です。() 内は、中心の X, Y, 半径, 開始角度, 終了角度です。Math.PI は円周率 (π : 3.141592...) を得る命令です。少し面倒ですが角度はラジアン単位で指定します。0.5*Math.PI で 90 度、1*Math.PI で 180 度、1.5*Math.PI で 270 度、2*Math.PI で 360 度です。



⑪ `context.fillStyle = '#0000FF';`

図形の塗りつぶし色を指定する `fillStyle()` です。色は 16 進数コードで指定します。

⑫ `context.fill();`

⑪で指定した色で図形を実際に塗りつぶす `fill()` です。

⑬ `context.fillText('Hello!', 50, 360);`

文字を描画する `fillText()` です。()内は、描画する文字列, X, Y です。

文字の輪郭だけを描く `strokeText()` というものもあります。



コラム⑥ Canvas は画像編集ソフト？

この授業では Canvas のほんの一部の機能しか紹介していません。簡単な図形や文字を描画したり、画像を表示したりする程度です。しかし Canvas という部品には、より複雑な図形を描画したり、画像を変形させたり回転させたり合成したりなど、まるで画像編集ソフトのような機能がたくさん備わっています。本格的に使いたい場合は自分で調べましょう。

✈ 3.2 タッチでお絵かき

	PC	And	iOS
Firefox	×	○	○
Chrome	×	○	○

スマホといえばタッチパネルが特徴です。タッチの情報を取得して、Canvas 上に指でお絵かきできるアプリを作りながら、タッチイベントの使い方を学びます。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 3.2 の中に、3.1 で作った index.html をコピーします。
- この index.html をもとに、<title>と<script>内を以下のように書きかえましょう。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 3.2 ¥ index.html

(略)

```
<head>
```

(略)

```
<title>3.2 タッチでお絵かき</title>
```

(略)

```
</head>
```

```
<body>
```

```
<!-- Canvas (画用紙・お絵かきセット) を置く -->
```

```
<canvas width='300' height='400' id='canvas'></canvas>
```

```
<script>
```

```
// Canvas を使うためのお決まりの 2 行
```

```
const canvas = document.getElementById('canvas'); // お絵かきセットを得て
```

```
const context = canvas.getContext('2d'); // その筆を得る
```

```
// タッチで描くための 3 個のイベント (開始、描画、終了)
```

①

```
canvas.addEventListener('touchstart', drawStart); // タッチ開始
```

```
canvas.addEventListener('touchmove', draw); // タッチ中の動き
```

```
canvas.addEventListener('touchend', drawEnd); // タッチ終了
```

```
// 各種変数の宣言
```

```
let touching = false; // タッチ中か否かの変数
```

```
let startX = 0; // 描画開始の X 座標
```

```
let startY = 0; // 描画開始の Y 座標
```

```
const penColor = '#000000'; // 筆の色
```

```
const penWidth = 5; // 筆の太さ
```

```
// 描画開始の関数 drawStart
```

```
function drawStart(event) {
```

②

```
touching = true;
```

```
// 指が触れた座標 (開始座標) を得る
```

```
startX = event.touches[0].pageX - canvas.offsetLeft;
```

③

```
startY = event.touches[0].pageY - canvas.offsetTop;
```

```
}
```



```
// 描画の関数 draw
function draw(event) {
  event.preventDefault(); // 画面をスクロールさせないようにする ④
  if (touching == true) { // タッチ中なら描画
    // 指が動いた後の座標（終了座標）を得る
    const endX = event.touches[0].pageX - canvas.offsetLeft; ⑤
    const endY = event.touches[0].pageY - canvas.offsetTop;
    // 開始座標から終了座標に向かって線を引く ⑥
    context.beginPath();
    context.moveTo(startX, startY); // 開始座標から
    context.lineTo(endX, endY); // 終了座標へ
    context.strokeStyle = penColor; // 筆の色
    context.lineWidth = penWidth; // 筆の太さ
    context.lineCap = 'round'; // 線の端を丸める ⑦
    context.stroke(); // 線を引く
    // 今の終了座標を次の開始座標にする ⑧
    startX = endX;
    startY = endY;
  }
}
```

```
// 描画終了の関数 drawEnd
function drawEnd(event) {
  touching = false;
}
```

</script>

（以下略）



スマホで動作確認

- スマホの **Firefox** で確認
 - アドレス例： **10.11.123.45/3.2/**
 - 指で自由に落書きしてみましょう。使える色は黒のみです。反応する指は1本だけです。
 - タッチパネルの性能・感度・個体差によっては書きにくいことがあります。
 - このプログラムはPC（マウス）では動きません（PC版は付録4参照）。



※ 紙面の都合で横倒しに表示しています



解説

① // タッチで描くための3つのイベント（開始、描画、終了）

画面に指が触れた情報を得るには、`touchstart`（指が触れた時）、`touchmove`（指が触れた状態で動いた時）、`touchend`（指が離れた時）の3つのイベントを設定します。マルチタッチに対応していて、触れている複数の場所の座標が一気に取れます（この授業では1ヶ所しか使いません）。

② `function drawStart(event) { ... }`

`touchstart` イベントに対応した関数です。（）内の `event` は引数（ひきすう）と言って、`touchstart` イベント発生時にこの関数に自動的に引き渡されます。この `event` オブジェクトの中に、触れた場所などの情報が入っています。`touch` 関連のイベントを使う場合は、それに対応した関数には必ずこの引数を書きます（名前は自由で、「event」でなくとも構いません）。

③ `startX = event.touches[0].pageX - canvas.offsetLeft;`

タッチされた場所の座標を得ています。

`event` オブジェクトには複数の指の情報が入っています。最初に触れた指の情報は `touches[0]` で得ることができます※。さらに `pageX` や `pageY` と指定することで、タッチされた場所（ブラウザのページ上の座標）を得ています。

`pageX` や `pageY` を得るだけでも良いのですが、Canvas に何かを描く時には、`canvas` 要素上の座標で指定します。`canvas` 要素はページの左上端にあるわけではなく、少し右・少し下にズレて置かれています。このズレ（オフセット）の量を `canvas.offsetLeft`（左側のズレ）や `offsetTop`（上側のズレ）で得て、引き算することで、`canvas` 要素上の座標に変換しています。

※ 2本目以降の指の情報は `touches[1]`, `touches[2]`, `touches [3]...` で得ることができます。

④ `event.preventDefault();`

この命令は「event のデフォルト（通常）の機能を使わない」という意味です。

`touchmove`、つまりタッチした状態で指を動かしたら、通常のアプリはどうなりますか？

そう、スクロールしますね。それが `touchmove` のデフォルトの機能です。しかし今回は Canvas 上で指を動かして落書きしようとしています。スクロールしては困ります。そこでこの命令で、指を動かしてもスクロールしないようにしています。スマホ特有のひと手間です。

⑤ `const endX = event.touches[0].pageX - canvas.offsetLeft;`

指が動いた後の座標を得ています。③と同様、`canvas` 要素のオフセットを引き算して `canvas` 要素上での座標に変換しています。

⑥ // 開始座標から終了座標に向かって線を引く

指が触れ始めた座標(`startX`, `startY`)から、指が動いた後の座標(`endX`, `endY`)に向かって線を引いています。指が動くたびにこれを繰り返すことで落書き機能を実現しています。

⑦ context.lineCap = 'round';

3.1 には無かった、線の端の形を指定する命令です。round というキーワードを指定することで、線の端を丸くしています。何も指定しないと端が角ばった線になります。角ばっていても機能上は問題ありませんが、丸くしたほうが描いた線がスムーズにつながります。

⑧ // 今の終了座標を次の開始座標にする

⑥の解説のようにこのアプリでは、指が動くたびに開始座標から終了座標に向かって線を引いています。線を引き終わったら、現在の終了座標を次に動いた時の開始座標にする、というテクニックです。

**コラム⑦ スマホのスクリーンショット**

スマホ画面をメモ代わりに画像として取り込むスクリーンショット（スクショ）は便利なものです。しかし、iPhone ユーザは Android のスクショ方法を知らない、その逆もまたしかり、ですよ。

iPhone： 電源ボタンとホームボタンの同時押し（長押しは強制再起動なので注意）

Android： 電源ボタンと音量を下げるボタンの同時長押し

撮ったスクショ画像は、iPhone だと写真と同じ DCIM フォルダに、Android だと Pictures フォルダの中の Screenshots フォルダに保存されます（写真は DCIM フォルダ）（バージョンによっても異なる場合があります）。

✧ 3.3 写真を撮る・見る

	PC	And	iOS
Firefox	○	○	○
Chrome	○	○	○

スマホの標準カメラアプリで写真を撮ったり、カメラロールの写真を選んで表示します。ローカル（スマホ内）にあるファイルを Web アプリで扱う方法を紹介します。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 3.3 の中に、3.1 または 3.2 で作った index.html をコピーします。
- この index.html をもとに、<title>と<body>内を以下のように書きかえましょう。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 3.3 ¥ index.html

(略)

```
<head>
```

(略)

```
<title>3.3 写真を撮る・見る</title>
```

(略)

```
</head>
```

```
<body>
```

```
<!-- ファイル選択ダイアログを使う<input>、カメラアプリも同時に使える -->
```

```
<input type='file' accept='image/*' id='fileChooser'>
```

①

```
<!-- Canvas (お絵描きセット) を置く -->
```

```
<canvas width='300' height='400' id='canvas'></canvas>
```

```
<script>
```

```
// Canvas を使うためのお決まりの 2 行
```

```
const canvas = document.getElementById('canvas'); // お絵描きセットを得て
```

```
const context = canvas.getContext('2d'); // その筆を得る
```

```
const fileChooser = document.getElementById('fileChooser');
```

```
const fileReader = new FileReader(); // ファイルを読む FileReader()
```

②

```
const img = new Image(); // 画像を入れる Image()
```

③

```
// イベント：ファイル選択ダイアログが変化したら readImage を実行
```

```
fileChooser.addEventListener('change', readImage);
```

④

```
// イベント：fileReader がファイル読み込みを完了したら loadImage を実行
```

```
fileReader.addEventListener('load', loadImage);
```

⑤

```
// イベント：img に画像がロードできたら drawImage を実行
```

```
img.addEventListener('load', drawImage);
```

⑥

```
// 画像を読み込む readImage 関数
```

```
function readImage() {
```

```
    const file = fileChooser.files[0];
```

⑦

```
    fileReader.readAsDataURL(file); // ファイルを URL 形式で読む
```

⑧

```
}
```

```
// Image オブジェクトの src に画像を入れる loadImage 関数
function loadImage() {
    img.src = FileReader.result;    // FileReader の result に画像がある ⑨
}

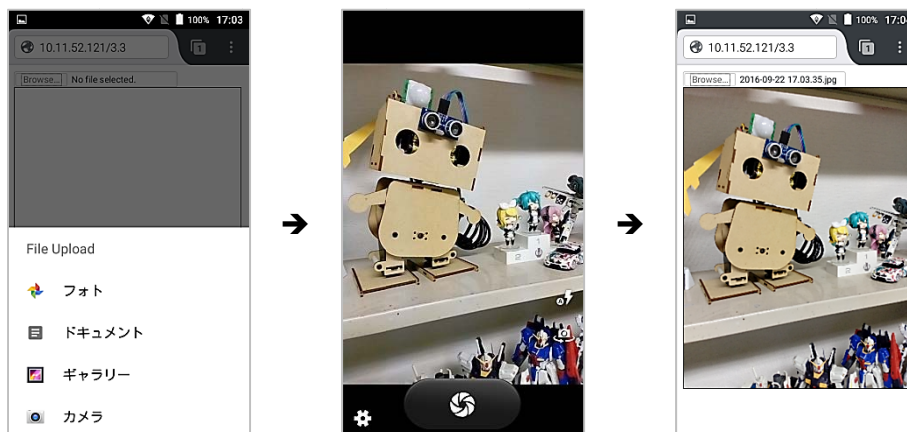
// Canvas に画像を置く drawImage 関数
function drawImage() {
    context.drawImage(img, 0, 0, 300, 400); // (Image,X,Y,W,H) ⑩
}
</script>
```

(略)



スマホで動作確認

- スマホの **Firefox** で確認
 - アドレス例: **10.11.123.45/3.3/**
 - **Browse...** をタップすると、[カメラ] や [ドキュメント] が選べます。
 - [カメラ] を選ぶと標準カメラアプリが起動し、撮影後に ☒ をタップすると、撮った写真が canvas 上に表示されます。
 - [ドキュメント] を選ぶと画像フォルダ (いわゆるカメラロール) から写真や画像を選べます。選んだ写真や画像が canvas 上に表示されます。



解説

① `<input type='file' accept='image/*' id='fileChooser'>`

`<input>` タグの `type` に `file` を指定すると、ローカル (PC やスマホ内) のファイルを選ぶダイアログを使えます。スマホの場合、標準カメラアプリとカメラロールにアクセスできます。`accept` 属性には、選択可能なファイル形式を指定します。`image/jpg` や `image/png` のように指定しますが、今回は `image/*` として、あらゆる画像ファイルを選べるようにしています。

② `const fileReader = new FileReader();`

①で選んだファイルを読み込むための読み込み屋さん、FileReader オブジェクトです。

③ `const img = new Image();`

読み込んだ画像を入れておく箱、Image オブジェクトです。1.6 の世界時計では要素に画像を読み込みましたが、要素と同じように画像を入れておける部品です。

④ `fileChooser.addEventListener('change', readImage);`

①でファイルが選択されると（写真が撮影されると）<input>要素は `change` イベントを発生させます。つまり、何らかのファイルが選択されたら `readImage` 関数を実行します。

⑤ `fileReader.addEventListener('load', loadImage);`

FileReader オブジェクトは、ファイルの読み込みが完了すると `load` イベントを発生させます。読み込みが完了したら `loadImage` 関数を実行します。

⑥ `img.addEventListener('load', drawImage);`

Image オブジェクトは、中に画像が入り終わると `load` イベントを発生させます。画像が入ったら `drawImage` 関数を実行します。

ここまでをまとめると、ファイルの選択～表示は以下の流れで行います。面倒ではありますが、定番の流れなので、深く考えずにこのまま使っていきましょう。

④ファイル選択完了 → `change` イベント発生 → `readImage` 実行（読み込む）
→ ⑤読み込み完了 → `load` イベント発生 → `loadImage` 実行（画像を入れる）
→ ⑥画像が入り終わる → `load` イベント発生 → `drawImage` 実行（表示する）

⑦ `const file = fileChooser.files[0];`

ファイル選択ダイアログで選んだファイルの名前 `file[0]` を `file` という変数に入れます。なぜファイル名が `file[0]` のように配列かというと、ファイル選択ダイアログは設定次第で複数のファイルを同時に選べるからです。今回は複数選択できる設定にしていますが、「選ばれた最初のファイル」ということで `file[0]` を指定します。

⑧ `fileReader.readAsDataURL(file);`

FileReader に `file (⑦)` という名前のファイルを読み込みさせる命令です。`readAsDataURL` は「URL 形式で読み込め」という意味です。URL 形式って？…ですが、「そういうものだ」と思って深く考えずに進みましょう。

⑨ `img.src = fileReader.result;`

FileReader が読み込んだファイルを Image オブジェクトの `src` の中に格納する処理です。FileReader が読み込んだファイルは `FileReader.result` に入っています。

⑩ `context.drawImage(img, 0, 0, 300, 400);`

`drawImage()`は Canvas 上に画像を置く命令です。()内は、画像 (Image オブジェクト)、画像を置く左上の X, Y, 幅(Width), 高さ(Height)です。

スマホのカメラで撮られた写真の縦横比はたいてい 16:9 ですが、今回は 400:300 つまり 4:3 にして表示しています。そのため、歪んで表示されるかもしれません。



コラム⑧ オブジェクトとは？

これまで「オブジェクト」について特に説明しませんでした。3.3 では FileReader オブジェクトと Image オブジェクト、3.2 では event オブジェクト、第1回では Date オブジェクトなどを使いました。オブジェクトって、何でしょう？

説明すると難しくなるのですが… **「部品である」**と理解しておいてください。

あれ？ 例えば

全部オブジェクトなのですが、慣例的に、見えない部品のことを「オブジェクト」と呼び、見える部品のことを「要素」と呼んでいます。例えば 3.3 で出た Image オブジェクトは、src 属性に画像を入れておく箱です。一方、要素も同様に src 属性に画像を入れておけます。違いは、Image オブジェクトは入れておくだけで見えず、要素には表示機能もあって見えます。

要素に画像を入れると画像は常に表示されていますが、画像をユーザに見せずに裏側で処理しただけのこともあります。見せたい画像は要素に入れ、見せる必要のない画像は Image オブジェクトに入れる、という使い分けをします（実際は、要素の画像を非表示にすることもできるのですが…）。

HTML や JavaScript だけでなく、「オブジェクト指向」と呼ばれる言語（Java, C++, C#, Visual Basic, Python, Ruby, Perl, PHP, Swift, COBOL など）は全て、そういう仕組みです。

✧ 3.4 写真に落書き

	PC	And	iOS
Firefox	×	○	○
Chrome	×	○	○

3.1～3.3のテクニックを組み合わせ、撮った写真やカメラロールにある写真にタッチで落書きして、さらにそれをローカル（スマホ内）に保存できるアプリを作ります。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 3.4 の中に、3.3 で作った index.html をコピーします。
- この index.html をもとに、<title>と<body>内を以下のように書きかえましょう。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 3.4 ¥ index.html

（略）

```
<title>3.4 写真に落書き</title>
```

（略）

```
<body>
```

```
<!-- ファイル選択ダイアログを使う<input>、カメラアプリも同時に使える -->
```

```
<input type='file' accept='image/*' id='fileChooser'>
```

```
<!-- Canvas（お絵描きセット）を置く -->
```

```
<canvas width='300' height='400' id='canvas'></canvas>
```

```
<!-- 色選択ダイアログ -->
```

```
<input type='color' value='#000000' id='col'>
```

```
<!-- 保存ボタン -->
```

```
<input type='button' value='保存（別窓に表示）' id='btn'>
```

```
<script>
```

```
// Canvas を使うためのお決まりの2行
```

```
const canvas = document.getElementById('canvas'); // お絵描きセットを得て
```

```
const context = canvas.getContext('2d'); // その筆を得る
```

```
const fileChooser = document.getElementById('fileChooser');
```

```
const fileReader = new FileReader(); // ファイルを読む FileReader()オブジェクト
```

```
const img = new Image(); // 画像を入れる Image()オブジェクト
```

```
// イベント：ファイル選択ダイアログが変化したら readImage を実行
```

```
fileChooser.addEventListener('change', readImage);
```

```
// イベント：fileReader がファイル読み込みを完了したら loadImage を実行
```

```
fileReader.addEventListener('load', loadImage);
```

```
// イベント：img に画像がロードできたら drawImage を実行
```

```
img.addEventListener('load', drawImage);
```

```
// 画像を読み込む readImage 関数
```

```
function readImage() {
```

```
    const file = fileChooser.files[0];
```

```
    fileReader.readAsDataURL(file); // ファイルを URL 形式で読む
```

```
}
```



```
// Image オブジェクトの src に画像を入れる loadImage 関数
function loadImage() {
    img.src = fileReader.result;    // fileReader の result に画像が入っている
}

// Canvas に画像を置く drawImage 関数
function drawImage() {
    context.drawImage(img, 0, 0, 300, 400); // (Image,X,Y,W,H)
}

// ここからは 3.2 の①以降をコピペ ( canvas 上にタッチでお絵かき ) ①
// タッチで描くための 3 個のイベント ( 開始、描画、終了 )
canvas.addEventListener('touchstart', drawStart);
canvas.addEventListener('touchmove', draw);
canvas.addEventListener('touchend', drawEnd);
// 各種変数の宣言
let touching = false;           // タッチ中か否かの変数
let startX = 0;                 // 描画開始の X 座標
let startY = 0;                 // 描画開始の Y 座標
let penColor = '#000000';      // 筆の色 ①
const penWidth = 5;            // 筆の太さ
// 描画開始の関数
function drawStart(event) {
    touching = true;
    // 指が触れた座標 ( 開始座標 ) を得る
    startX = event.touches[0].pageX - canvas.offsetLeft;
    startY = event.touches[0].pageY - canvas.offsetTop;
}
// 描画の関数
function draw(event) {
    event.preventDefault();      // 画面をスクロールしないようにする
    if (touching == true) {      // タッチ中なら描画
        // 指が動いた後の座標 ( 終了座標 ) を得る
        const endX = event.touches[0].pageX - canvas.offsetLeft;
        const endY = event.touches[0].pageY - canvas.offsetTop;
        // 開始座標から終了座標に向かって線を引く
        context.beginPath();
        context.moveTo(startX, startY); // 開始座標から
        context.lineTo(endX, endY);     // 終了座標へ
        context.strokeStyle = penColor; // 筆の色
        context.lineWidth = penWidth;   // 筆の太さ
        context.lineCap = 'round';      // 線の端を丸める
        context.stroke();                // 線を引く
        // 今の終了座標を次の開始座標にする
        startX = endX;
        startY = endY;
    }
}
}
```

```
// 描画終了の関数
function drawEnd(event) {
    touching = false;
}
```

// 3.2 のコピペここまで

①

```
// 色の設定
const col = document.getElementById('col');
col.addEventListener('change', changeColor);
// 筆の色を変える changeColor 関数
function changeColor() {
    penColor = col.value;
}
```

②

//保存ボタンの処理

③

```
const btn = document.getElementById('btn');
btn.addEventListener('click', saveImage);
// 画像を保存する（別窓に表示する）saveImage 関数
function saveImage() {
    const imgWin = window.open('', '');
    imgWin.document.write('');
    imgWin.document.write('<div>画像を長押しして保存できます</div>');
    imgWin.document.close();
}
```

④

⑤

⑥

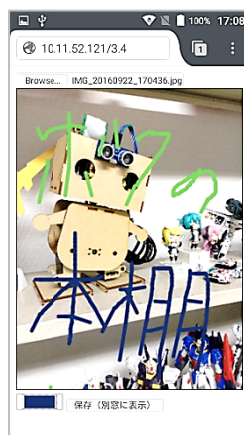
</script>

（略）



スマホで動作確認

- スマホの **Firefox** で確認
 - アドレス例： **10.11.123.45/3.4/**
 - 3.3 の写真選択機能に加え、3.2 の落書き機能が付きました。
 - 色選択ダイアログで筆の色も変えられます。
 - 保存（別窓に表示） を押すと別窓に画像が出て、画像を長押しすると保存できます。





解説

① //ここからは3.2の①以降をコピペ（canvas上にタッチでお絵かき）

3.2のindex.htmlを開き、①（3個のイベントの記述）のところから、drawEnd関数の終わりまでを、まるごとコピペします。

ただし、筆の色を変えるプログラムを追加するので、`var penColor = '#000000'`は`let penColor = '#000000'`として、penColorを再代入可能な変数に変更します。

② //色の設定

色選択ダイアログです。これでpenColorの色を変えています。

③ //保存ボタンの処理

保存（別窓に表示） ボタンを押すと saveImage 関数を実行します。

④ const imgWin = window.open('', '');

window.open()は新しいウィンドウ（タブ）を作る命令です。（）内に2個の引数を指定します。1個目はURLで、作ったウィンドウに何らかのWebページを表示したい場合はURLを指定します。2個目はこのwindowオブジェクトの名前ですが、通常は指定しなくても構いません。ここでは2個とも空白文字列にしています。

⑤ imgWin.document.write('');

新しく作ったウィンドウimgWinのdocument（ページ）に（）内のHTML要素を書き込む命令です。要素を作り、そのsrcにcanvasの中身（写真と落書き）のデータを指定しています。toDataURL()というのは「URL形式のデータ」という意味ですが、深く考えず「そうやるものだ」と思っておきましょう。シングルとダブルのクォーテーションが入り乱れていますが、どこからどこまでがセットか、注意しながら入力しましょう。

⑥ imgWin.document.close();

documentオブジェクトを閉じています。④open、⑤write、⑥closeはセットで使います。

④～⑥で、新しいウィンドウにcanvas要素の中身を画像として置けました。あとは、ブラウザが元々持っている機能を使って、画像を長押しすると保存やコピーできる、というわけです。



コラム⑨ コピペも大事、ただし…

3.4のコードは長いですが、ほとんど3.3と3.2のコピペです。レポートなどでは「コピペはNG」ですが、プログラミングではコピペを活用しましょう。短い（低機能な）プログラムをいくつか作り、コピペで組み合わせて長い（高機能な）プログラムを作るのは常套手段です。また、うまく動かない時には世にあるサンプルをコピペして動かしながら、不具合の原因究明をしたりします。ただし授業では、自分が書いたコードのコピペに限りましょう。ラクする目的で人が書いたコードをコピペしていると、全く身につかないので、履修する意味がなくなります。

✈ 3.5 動画と音声

	PC	And	iOS
Firefox	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Chrome	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

つい最近まで Web アプリで動画や音声を扱うには Flash を使う必要がありました。今は、HTML だけで動画や音声を扱えます。超簡単なメディアプレイヤーアプリを作ります。



HTML を入力

- C: ¥ xampp ¥ htdocs ¥ 3.5 の中に、1.1 で作った index.html (テンプレート) をコピーします。
- この index.html をもとに、以下のように書き加えましょう。
- CampusSquare から「PHC_03_素材」をダウンロードし、3.5 素材フォルダの中にある 'robot.mp4' (動画) と 'yarujanaika.mp3' (音声) を 3.5 フォルダに入れましょう。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 3.5 ¥ index.html

(略)

```
<head>
  (略)
  <title>3.5 動画と音声</title>
</head>
<body>
  <div id='msgVideo'>video タグの例</div>
  <video src='robot.mp4' controls></video> ①
  <div id='msgAudio'>audio タグの例</div>
  <audio src='yarujanaika.mp3' controls></audio> ②
</body>
</html>
```



スマホで動作確認

- スマホの  **Firefox** で確認
 - アドレス例: 10.11.123.45/3.5/
 - 動画と音声再生できましたか?





解説

① `<video src='robot.mp4' controls></video>`

ページに動画ファイルを置く`<video>`タグです。src 属性に動画ファイル名を指定します。controls 属性を付けると、再生/停止ボタンや再生位置スライダー、音量コントロールなどを表示します。iOS 端末では動画は自動的に全画面で再生されます。

再生できるファイル形式はブラウザごとにまちまちで（特許権など大人の事情）今後も変わっていきそうです。現状、mp4(H.264)、webm、ogv 形式が主流です。必要に応じて調べましょう。

② `<audio src='yarujanaika.mp3' controls></audio>`

ページに音声ファイルを置く`<audio>`タグです。src 属性に音声ファイル名を指定します。controls 属性は`<video>`タグと同じです。

音声ファイルもブラウザごとに対応がまちまちで、現状、mp3、wav、ogg 形式が主流です。

超簡単ですね。サーバにある動画を配信するだけなら、たった 1 行でメディアプレイヤーができました。その他にも、`<audio>`タグに音声ファイルを読み込んでおいて（controls を付けないで）ユーザの操作に効果音を付けることなどもできます。ゲームなどにも活用できそうですね。必要に応じて発展した使い方を調べてみましょう。



コラム⑩ サンプル動画のロボットは？

サンプル動画のロボットは「ピッコロボ」といいます。

<https://www.vstone.co.jp/products/piccorobo/>

二足歩行ロボットとしては破格の 1 万円のキットです。制御はアルドゥイーノというマイコンで行っていて、プログラミングがとても簡単です。センサを追加して自律歩行させたり、通信機能を追加してスマホで操縦したりできます。

この授業でも中盤以降で Arduino を使います。ピッコロボのような「ロボット」は作りませんが、スキルは同じです。PC（伝統的なコンピュータ）だけでなく、Web、スマホ、マイコンという新時代のコンピューティングのスキルも身に付けることで、皆さんの将来の可能性が大いに広がります！

※ このサンプル動画は國分が自宅で撮影しました（笑）

✧ 3.6 My カメラアプリ

	PC	And	iOS
Firefox	○	○	×
Chrome	△	△	×

標準カメラアプリを使わずに、カメラデバイスに直接アクセスして映像を取り込む方法を紹介します。カメラを用いたオリジナルアプリを作る基礎を学びます。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 3.6 の中に、1.1 で作った index.html (テンプレート) をコピーします。
- この index.html をもとに、以下のように書き加えましょう。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 3.6 ¥ index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <meta name='viewport' content='width=device-width,initial-scale=1'>
    <title>3.6 My カメラアプリ</title>
  </head>
  <body>
    <video width='300' autoplay id='video'></video> ①
    <canvas width='300' height='400' id='canvas'></canvas>
    <script>
      const video = document.getElementById('video');

      // カメラとマイクを使う命令 navigator.mediaDevices.getUserMedia()
      const media = navigator.mediaDevices.getUserMedia( ②
        {
          video: true,    // カメラを使うか否か ③
          audio: false    // マイクを使うか否か
        }
      );
      // メディアストリームの取得に成功したら onSuccess、失敗したら onError を実行
      media.then(onSuccess, onError); ④

      // 成功時の onSuccess 関数
      function onSuccess(stream) {    // カメラ映像は stream に入ってくる ⑤
        // video タグの src に、stream を URL 形式にしたものを入れる
        video.src = window.URL.createObjectURL(stream); ⑥
      }

      // エラー時の onError 関数
      function onError(err) {         // エラー内容が err に入ってくる
        window.alert(err);           // 警告ダイアログに err を表示
      }
    </script>
  </body>
</html>
```

```
// 映像クリックで Canvas にキャプチャ
video.addEventListener('click', capImage);
const canvas = document.getElementById('canvas');
const context = canvas.getContext('2d');
// Canvas に画像を取り込む capImage 関数
function capImage() {
    context.drawImage(video, 0, 0, 300, 400); // (video,X,Y,W,H) ⑦
    // キャプチャした画像に日時を書く
    const now = new Date(); ⑧
    const str = now.getFullYear() + '/'
        + now.getMonth() + '/'
        + now.getDate() + ' '
        + now.getHours() + ':'
        + now.getMinutes() + ':'
        + now.getSeconds();
    context.fillStyle = '#FFFFFF'; // 塗りつぶしの色
    context.font = '28px sans-serif'; // サイズ、書体
    context.fillText(str, 8, 30); // fillText(text,X,Y)
    context.strokeStyle = '#000000' // テキストの外枠
    context.strokeText(str, 8, 30); // strokeText(text,X,Y)
}
</script>
</body>
</html>
```



スマホで動作確認

● スマホの **Firefox** で確認※¹

- アドレス例： **10.11.123.45/3.6/**
- 「カメラを共有しますか？」と出たら、使いたいカメラ（バック／フロント）を選んでから、**共有**をタップ
- 選んだカメラの映像が表示されます。映像をタップすると静止画として下方に表示されます。静止画には日付と時刻が入っています。



※ 紙面の都合で横倒しに表示しています

※¹ Chrome の場合バージョンや条件により動作したりしなかったりします。



解説

① `<video width='300' autoplay id='video'></video>`

3.5 で出た `<video>` タグですが、今回は動画ファイルを見るためではなく、カメラの映像を見るために使います。`<video>` タグは動画ファイルに限らず、様々な映像ストリームを表示できます。

`width` 属性を指定して見やすい大きさにしています。また、`autoplay` 属性（自動再生）を指定しています。動画なら自分で再生／停止できるほうが便利ですが、今回はカメラ映像を見るのが目的ですから、起動したらすぐに見られるように自動再生を設定しました。

② `const media = navigator.mediaDevices.getUserMedia(...);`

カメラとマイク（メディアデバイス）を使うための命令 `getUserMedia` の定義です。`getUserMedia` は現状、ブラウザやそのバージョンごとに命令の名前が統一されていないため、このままでは使えないブラウザもあります。Firefox は問題なく動作します（iOS は除く）。

③ `{ video: true, audio: false }`

カメラとマイク、それぞれを使うか否かの設定です。今回はカメラのみ使います。マイクで音声もプレビューする場合は `audio` も `true` にします。

④ `media.then(onSuccess, onError);`

ここでメディアデバイスのプレビューを開始しています。（ ）内には 2 個の引数を指定します。

1 個目： デバイスへのアクセスに成功した時に実行する関数

2 個目： デバイスへのアクセスに失敗した時に実行する関数

⑤ `function onSuccess(stream) { ... }`

④ が成功した時に実行される関数です。`stream` オブジェクトが引数に入ります。`stream` オブジェクトの中に、カメラやマイクのデータ（映像や音声）が入ります（今回はカメラのみ）。

⑥ `video.src = window.URL.createObjectURL(stream);`

`<video>` 要素の `src` に `stream` を入れています。この際、`stream` を URL 形式として入れています。「そういうものだ」とっておいてください。

⑦ `context.drawImage(video, 0, 0, 300, 400);`

3.3 で使った、Canvas に画像を置く `drawImage()` です。Canvas には画像（静止画）だけでなく動画も指定できます。カメラのプレビュー（`<video>` 要素）をタップすると、Canvas 上にその瞬間の静止画が取り込まれて表示されます。

⑧ `const now = new Date();` 以降

第 1 回の時計アプリで紹介したテクニックで「年/月/日 時:分:秒」という文字列を作り、3.1 で紹介した `fillText()` と `strokeText()` で canvas に表示しています。

<補足>

標準カメラアプリ(3.3)でも写真は撮れますが、シャッターを押さないといけませんし、写真を加工しようとするとは別のアプリを使う必要があります。今回もタップで撮影してはいますが、日時を描くなど、自分のアプリの中で加工しています。さらに、例えばタイマーを使って定期的に撮影したり、スマホが強い衝撃を受けたら自動的に撮影したり、ビデオチャットアプリやAR(拡張現実)アプリなど、様々な可能性が広がります。ただ、navigator.mediaDevices.getUserMedia()は仕様が固まっておらず、今後とも変化していきそうです。必要に応じてウォッチしていきましょう。



コラム⑩ プログラミング言語は英語？

プログラミングでは英語的な言葉を使うため「英語の力が必要」と思われがちですが…

それは違います。

プログラミングはパズルや積み木に近く、言葉そのものを理解しなくてもできます。たいてい定番のパターン(パズルのピース、部品)があって、多くのパターンを知り、上手に組み合わせることが大事です。英語「的」ではありますが、パズルだと考えて、恐れずに学んでいきましょう。

ただ、英文を理解できると、より進んだことができるのは確かです。新しい仕様やテクニックの情報(仕様書やドキュメント)はたいてい英語です。ソフトウェアは世界中で使われますから、そうなります。英文が読めれば、日本ではほとんど知られていない新しいテクニックも使うことができます。幸いなことに、仕様書やドキュメントの英文はたいてい、平易に書かれています。翻訳ソフトなども使いながら、頑張って読みましょう。



コラム⑪ プログラミングは数学？

プログラミングでは数式を書くことがあるため「数学の力が必要」と思われがちですが…

それは違います。

確かに数式は使いますが、たいていは中学レベルの数式で間に合います。少し凝ったものを作ろうとすると高校レベルの数式も使いますが…。プログラミングの場合、正確でスマートな数式や方程式を使うことよりも、「動く」ことが重要です。自分が理解できる簡単な数式でいかにそれっぽくできるか、知識よりもアイデアが大事です。

ただ、ずっと逃げていてもいけません。例えばゲームのキャラクターをジャンプさせたり、何かに当たって跳ね返ったり、グルグル回ったり、落ちたり…。それっぽく動かそうとすると、やはりそれなりの数式は必要です。でもそれらにも定番のパターンがあるので、自分がやりたいことのパターンについては頑張って勉強しましょう。

✧ 3.7 音声合成アプリ

	PC	And	iOS
Firefox	△	×	×
Chrome	○	○	○

音声は自然で直感的なインタラクションに欠かせません。ブラウザの音声合成機能を使い、自由に入力したテキストを話すアプリを作ります。また、話した言葉をテキストに変換する音声認識の方法も紹介します。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 3.7 の中に、1.1 で作った index.html (テンプレート) をコピーします。
- この index.html をもとに、以下のように書き加えましょう。
- CampusSquare から「PHC_03_素材」をダウンロードし、3.7 素材フォルダの中にある 'image.jpg' を 3.7 フォルダに入れましょう。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 3.7 ¥ index.html

(略)

```
<title>3.7 音声合成アプリ</title>
<style>
  #txt { font-family: sans-serif;
        font-size: 24px;}
</style>
</head>
<body>
  <input type='text' value='こんにちは' id='txt'>
  <img src='image.jpg' id='img'>
  <script>
    // Web Speech Synthesis API を使う設定
    const msg = new SpeechSynthesisUtterance();
    msg.lang = 'ja-JP';      // 日本語に設定 (英語は en-US) ①
    msg.volume = 1.0;       // 音量設定 (0~1)
    msg.pitch = 1.5;        // ピッチ設定 (0~2)
    // 画像をクリックしたら話す (関数 speak を実行)
    const img = document.getElementById('img');
    img.addEventListener('click', speak);
    // 話す関数 speak
    function speak() {
      txt = document.getElementById('txt');
      msg.text = txt.value;  // テキストボックスの文章を設定
      window.speechSynthesis.speak(msg); // 実際に話させる命令 ②
    }
  </script>
</body>
</html>
```



スマホで動作確認

- スマホの **Chrome** で確認^{※1}
 - アドレス例： **10.11.123.45/3.7/**
 - 画像をタップすると、テキストボックスの文字列が読み上げられましたか？



※1 Firefox は対応していません。



解説

① `const msg = new SpeechSynthesisUtterance();`

音声合成の各種設定を行う `SpeechSynthesisUtterance` オブジェクトです。

`msg.lang = 'ja-JP';` ... 話す言語の設定です。日本語に設定しています。
`msg.volume = 1.0;` ... 話す音量の設定です。0~1で指定します。
`msg.pitch = 1.5;` ... 話すピッチ（速さ・高さ）の設定です。0~2で設定します。

② `window.speechSynthesis.speak(msg);`

ブラウザの音声合成機能を使って実際に発話させる命令です。①の設定で話します。

`msg.text = txt.value;` ... 話すテキストです。テキストボックスの内容を入れています。

音声合成と言うと難しそうですが、HTML と JavaScript を使うと割と簡単ですね。こんなに簡単なら、SNS のメッセージやニュースの文章を読み上げたり、ゲームのキャラクターに話させたり、いろいろと使えそうです。

自分のスマホでも試す際は、Android では、「テキスト読み上げエンジン (TextToSpeech; TTS)」を事前にインストールしておく必要があります（付録 5 参照）。iOS は TTS のインストールは不要です。



参考コード：音声認識アプリ

音声認識も割と簡単です。大学のネットワークでは動かないため授業では割愛しますが、自宅のWi-Fi環境やモバイルルーターがあって、興味があったら以下のコードを試してみましょう。ブラウザはChromeを使いましょう。

詳しくは自分で調べてみましょう。

参考：音声認識（提出不要） [index.html](#)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <meta name='viewport' content='width=device-width,initial-scale=1'>
    <title>3.7s 音声認識</title>
  </head>
  <body>
    <div id='msg'>ここに認識結果を表示</div>
    <script>
      const msg = document.getElementById('msg');
      // 音声認識の設定
      const recog = new webkitSpeechRecognition();
      recog.lang = 'ja-JP';          // 日本語として認識
      // イベント設定
      recog.addEventListener('soundstart', onStart);          // 認識開始
      recog.addEventListener('nomatch', onNoMatch);           // 認識失敗
      recog.addEventListener('result', onResult);              // 認識完了
      // 認識を開始した時の関数
      function onStart() {
        msg.innerHTML = '認識中...';
      }
      // 認識できなかった時の関数
      function onNoMatch() {
        msg.innerHTML = '認識できません';
      }
      // 認識を完了した時の関数
      function onResult(event) {
        msg.innerHTML = event.results[0][0].transcript;
      }
      // 音声認識開始
      recog.start();
    </script>
  </body>
</html>
```










スマホの画像を消す

- 授業内に撮った写真や保存した画像は、個人情報保護とスマホのメモリ節約のために、消してから帰りましょう。
 - － 「ギャラリー」アプリを起動して削除します
 - － 個人的に残しておきたい画像は、スマホを PC に接続して、各自の USB メモリなどに移動させて持ち帰ってください



データを持ち帰る

- 本日作成したデータは全て USB メモリなどにコピーして持ち帰りましょう
 - － C: ¥ xampp ¥htdocs 内の以下のフォルダ (7 個)
 - －  3.1  3.2  3.3  3.4  3.5  3.6  3.7



本日の課題

- ① 3.4、3.6 のコードを完成させる (可能ならスマホで動作確認する)
 - ・ 資料のコードをもとに自分なりのアレンジを加えても構いません。その際は②の readme.txt の所感にアレンジ内容を書いてください。効果的なアレンジであれば成績評価に加点します。
- ② readme.txt というテキストファイルを作り、以下を書く
 - ・ 学籍番号 と 氏名
 - ・ 所感 (考えたこと、感想、応用アイデアなど。字数不問だが数行は書いてほしい。)
 - ・ 質問 (無ければ不要)

【提出方法】

- ・ ①のフォルダ 2 個と、②の「readme.txt」1 個、全てをまとめて zip 圧縮
- ・ zip のファイル名は「学籍番号.zip」とする (例: 12345nhu.zip)
- ・ CampusSquare のレポート「フィジカルコンピューティング 03」から提出

【提出締切】 **10月20日(木) 15:00** (遅れてしまった場合は担当教員に相談のこと)