



第 4 回

スマホの 様々なセンサを扱う

スマホにはたくさんのセンサが入っています。
加速度、ジャイロ、照度、近接、GPSセンサを
HTMLとJavaScriptで使い倒します。

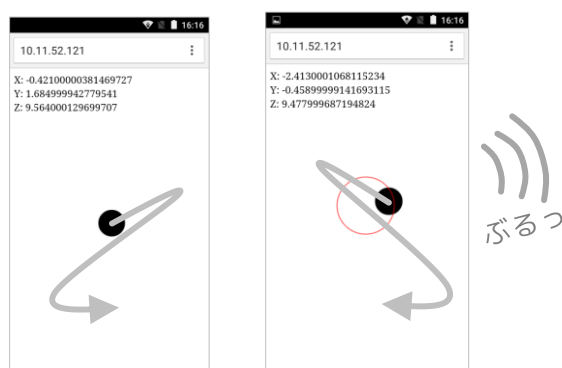
✧ 4.0 スマホの様々なセンサを扱う

スマホには、加速度センサ、ジャイロセンサ、照度センサ、近接センサ、GPS センサなどが入っています。PC には無いこれらのセンサをうまく使うと、ユーザや周囲の状態をもとに、便利なことや楽しいことが実現できます。フィジカルコンピューティングの道具であるスマホを最大限に使い倒すための方法を学びましょう。

4.1 加速度センサ

→ 3 ページ

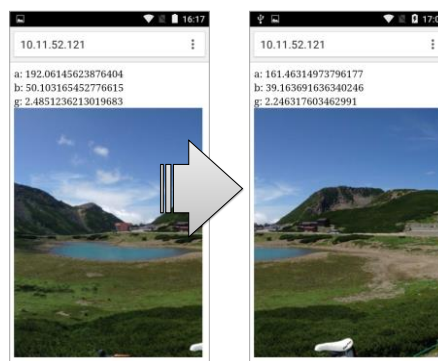
加速度センサは、スマホの左右、上下、表裏方向の動きを測ります。スマホを回すと画面が回転したり、万歩計アプリで歩数が測れるのはこのセンサのおかげです。三つのサンプルを通して加速度データの扱い方を学びます。



4.2 ジャイロセンサ

→ 9 ページ

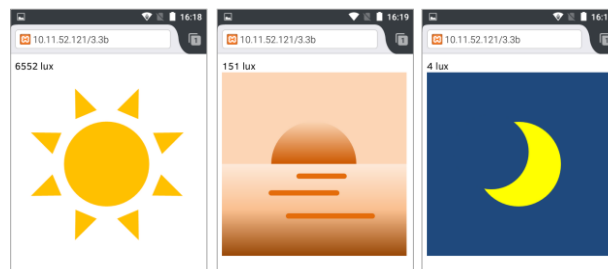
ジャイロセンサは、スマホの回転や、ユーザが向いている方位を測ります。パノラマ写真撮影や、空にかざして見るプラネタリウムアプリは、このセンサのおかげです。二つのサンプルを通してジャイロデータの扱い方を学びます。



4.3 照度センサ (Firefox 限)

→ 13 ページ

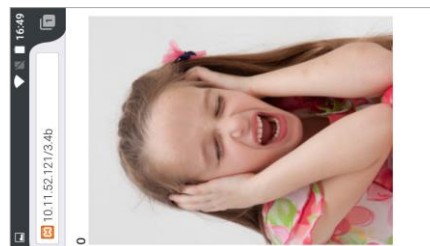
照度センサは、明るさを測ります。周りの明るさに応じて画面の明るさが変わったり、裏向きにすると自動でスリープさせたりできるのはこのセンサのおかげです。二つのサンプルを通して照度データの扱い方を学びます。



4.4 近接センサ (Firefox 限)

→ 17 ページ

近接センサは、ユーザとの距離を測ります (たいのスマホは近い / 遠い程度ですが) 。スマホを耳にあてて通話する時に画面がロックされるのはこのセンサのおかげです。二つのサンプルを通して近接データの扱い方を学びます。



4.5 GPS センサ

→ 21 ページ

GPS[※]センサは、人工衛星からの電波をもとにユーザがいる位置を測ります。地図アプリや道案内アプリ、天気アプリで自分の居場所が分かるのはこのセンサのおかげです。二つのサンプルを通して GPS データの扱い方を学びます。

※ Global Positioning System の略



本日の課題

→ 26 ページ

✈ 4.1a 加速度センサ

	PC	And	iOS
Firefox	x	○	○
Chrome	x	○	○

加速度センサは、スマホの左右、上下、表裏方向の動きを測ります。スマホを回すと画面が回転したり、万歩計アプリで歩数が測れるのはこのセンサのおかげです。三つのサンプル（4.1a/4.1b/4.1c）を通して加速度データの扱い方を学びます。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 4.1a 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。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 4.1a ¥ index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <meta name='viewport' content='width=device-width,initial-scale=1'>
    <title>4.1a 加速度センサ</title>
  </head>
  <body>
    <div id='msg'>ここにデータを表示</div>
    <script>
      // 加速度センサの値を得る devicemotion イベント
      window.addEventListener('devicemotion', getData);
      function getData(event) {
        // 重力加速度 accelerationIncludingGravity [m/s^2]
        // 地球の重力 (1G) は約 9.8 m/s^2
        const accX = event.accelerationIncludingGravity.x; // 左右 ②
        const accY = event.accelerationIncludingGravity.y; // 上下
        const accZ = event.accelerationIncludingGravity.z; // 表裏
        const msg = document.getElementById('msg'); // データ表示
        msg.innerHTML = 'X: ' + accX + '<BR>' +
          'Y: ' + accY + '<BR>' +
          'Z: ' + accZ;
      }
    </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

メモ
★

- スマホの **Chrome** で確認^{※1}

- メモしたアドレス (★) に続けて「/4.1a/」と入力して Enter
- アドレス例 : **10.11.123.45/4.1a/**
- 画面の自動回転を OFF にしておくほうが確認しやすい



※1 Firefox でも動作しますが、授業で使うスマホとの相性のため Chrome を使います。



解説

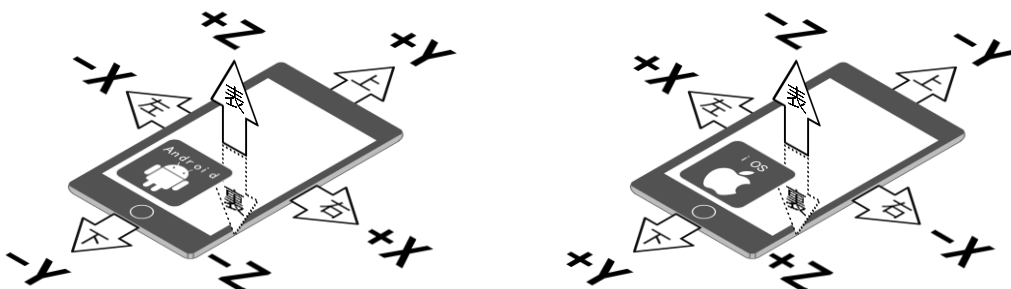
- ① **window.addEventListener('devicemotion', getData);**

加速度センサのデータが変化すると **devicemotion** イベントが起きます。これが起きたらデータを取り出す関数 (今回は **getData** と名付けた) を実行させます。

- ② **const accX = event.accelerationIncludingGravity.x;**

加速度データは、**accelerationIncludingGravity** (重力加速度という言葉) で取り出せます。自動車や電車が走り出す時は後ろに、止まる時は前に、曲がる時は横に体が持って行かれますが、それが加速度です。急に動く (速度が変化する) ほど強く感じます。単位は m/s^2 です。

下図のように、スマホの左右方向の加速度が **x**、上下方向が **y**、表裏方向が **z** で取り出せます。Android と iOS では値の + / - が逆です。



✈ 4.1b 加速度を見える化

	PC	And	iOS
Firefox	×	○	○
Chrome	×	○	○

加速度の数値だけを見ても分かりにくいので、具体的にどんな値なのか見えるように、加速度の大きさによってボールが動くアプリを作ります。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 4.1b の中に、4.1a で作った index.html 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。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 4.1b ¥ index.html

(略)

```

<title>4.1b 加速度を見える化</title>
</head>
<body>
  <div id='msg'>ここにデータを表示</div>
  <canvas width='300' height='300' id='canvas'></canvas>

  <script>
    // 加速度センサの値を得る devicemotion イベント
    window.addEventListener('devicemotion', getData);
    function getData(event) {
      // 重力加速度 accelerationIncludingGravity [m/s^2]
      // 地球の重力 (1G) は約 9.8 m/s^2
      const accX = event.accelerationIncludingGravity.x; // 左右
      const accY = event.accelerationIncludingGravity.y; // 上下
      const accZ = event.accelerationIncludingGravity.z; // 表裏
      const msg = document.getElementById('msg'); // データ表示
      msg.innerHTML = 'X: ' + accX + '<BR>' +
        'Y: ' + accY + '<BR>' +
        'Z: ' + accZ;
      drawBall(accX, accY, accZ); // ボールを描く ①
    }

    // ボール (円) を canvas に描く (スマホの自動回転を切っておくとよい)
    const canvas = document.getElementById('canvas');
    const context = canvas.getContext('2d');
    const cX = canvas.width / 2; // canvas の中心の X 座標
    const cY = canvas.height / 2; // canvas の中心の Y 座標
    const baseR = 50; // Z が 1G の時のボールの半径
    // canvas にボール (円) を描く関数 drawBall
    function drawBall(x, y, z) { ②
      // canvas の内容を消す clearRect()
      context.clearRect(0, 0, canvas.width, canvas.height); ③
    }
  </script>

```

```
// 円を描く ( arc の各引数の係数(15 や 3 など)は現物合わせのテキトー )
context.beginPath();
context.arc(cX-x*15, cY+y*15, baseR-z*3, 0, 2*Math.PI);
context.fillStyle = '#000000';
context.fill();
```

④

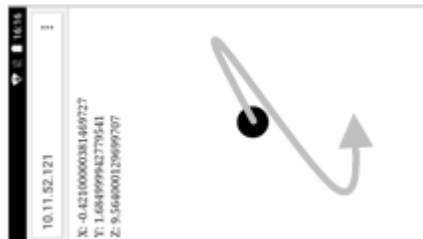
</script>

(以下略)



スマホで動作確認

- スマホの  **Chrome** で確認^{※1}
 - アドレス例 : 10.11.123.45/4.1b/



紙面の都合で横倒しに表示しています。

※1 Firefox でも動作しますが、授業で使うスマホとの相性のため Chrome を使います。



解説

① **drawBall(accX, accY, accZ);**

加速度のデータを drawBall という関数に渡しています。

② **function drawBall(x, y, z) { ... }**

加速度のデータをもとに、canvas にボール(円)を描く関数です。関数の宣言で、() 内に変数を書くと、他の処理からデータを渡すことができ、渡されたデータをこの関数の中で計算などに使えます。今回は、加速度の大きさによって円の位置を変えています。

③ **context.clearRect(0, 0, canvas.width, canvas.height);**

canvas の内容を消す clearRect です。()内は、X,Y,幅,高さです。canvas でアニメーションを実現するにはこのように、一旦 clearRect で消して、新しく描く、を繰り返します。

④ **context.arc(cX-x*15, cY+y*15, baseR-z*3, 0, 2*Math.PI);**

左右の加速度 x が + なら円の中心を左に / - なら右に、上下の加速度 y が + なら円の中心を下に / - なら上に、表裏の加速度 z が + なら円の半径を小さく / - なら大きくしています。なぜそうなるかというより、それっぽく見えるようにこうした、という一文です。

✈ 4.1c ゆらゆらゲーム

	PC	And	iOS
Firefox	x	○	○
Chrome	x	○	○

上下左右の加速度が一定の値以上になると「ぶるっ」とするバランスゲームです。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 4.1c の中に、4.1b で作った index.html 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。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 4.1c ¥ index.html

(略)

```

<title>4.1c ゆらゆらゲーム</title>
</head>
<body>
  <div id='msg'>ここにデータを表示</div>
  <canvas width='300' height='300' id='canvas'></canvas>
  <script>
    // 加速度センサの値を得る devicemotion イベント
    window.addEventListener('devicemotion', getData);
    function getData(event) {
      (データを取ってくる部分は変更ないので省略)
      drawBall(accX, accY, accZ); // ボールを描く
      alertVibration(accX, accY, 3); // 閾値(ここでは3)より大なら震わせる ①
    }
    // ボール(円)を canvas に描く(スマホの自動回転を切っておくとよい)
    (ここにある変数の定義は変更ないので省略)
    // canvas にボール(円)を描く関数 drawBall
    (drawBall 関数は変更ないので省略)
    // 震わせる(バイブレータを振動させる)
    function alertVibration(x, y, threshold) { ②
      // 加速度のスカラーを求める(三平方の定理)
      const scalar = Math.sqrt(Math.pow(x,2) + Math.pow(y,2)); ③
      if (scalar > threshold) { // スカラーが閾値より大きい時
        navigator.vibrate(100); // バイブレータを 100ms 振動 ④
      }
      // threshold(閾値)の円を描く
      context.beginPath();
      context.arc(cX, cY, threshold*15, 0, 2*Math.PI);
      context.strokeStyle = '#FF0000';
      context.stroke();
    }
  </script>

```

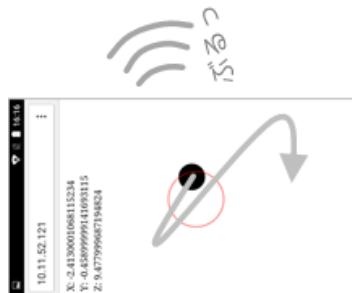
(以下略)



スマホで動作確認

● スマホの Chrome で確認※1

- アドレス例： **10.11.123.45/4.1c/**
- スマホを手のひらにのせ、目をつむったり、片足で立ったり、歩いたりして、ぶるっとしないようにバランスをとりましょう。コップの水をこぼさないようなイメージで。



紙面の都合で横倒しに表示しています。

※1 Firefox でも動作しますが、授業で使うスマホとの相性のため Chrome を使います。



解説

① `alertVibration(accX, accY, 3);`

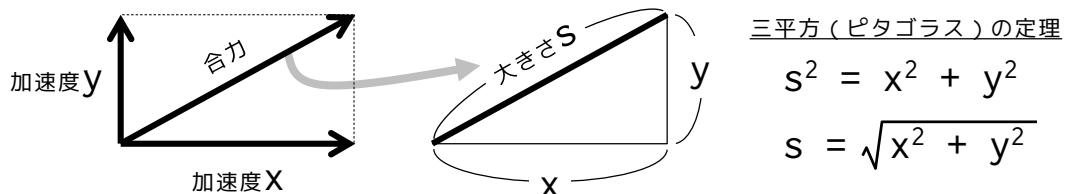
左右と上下の加速度データと、閾値（それより大きいなら震わす）を入れて `alertVibration` という関数②を実行しています。閾値を小さくするほどゲームは難しくなります。この例の「3」は自動車ではけっこう強めのブレーキやアクセル（不快なレベル）にあたる加速度です。

② `function alertVibration(x, y, threshold) { ... }`

加速度が一定の値より大きければバイブレータを振動させる関数の定義です。

③ `const scalar = Math.sqrt(Math.pow(x,2) + Math.pow(y,2));`

数式ですが、シャッターをおろさないでください！ 中学で学ぶ三平方（ピタゴラス）の定理で、左右加速度 x と上下加速度 y の合力の大きさ（スカラー）を求めています（下図）。`Math.sqrt()` は平方根（ルート）、`Math.pow(a,b)` はべき乗 a^b を求める関数です。



④ `navigator.vibrate(100);`

バイブレータを振動させる命令です。（）内に振動させる時間をミリ秒単位で指定します。この場合、100 ミリ秒 = 0.1 秒間、振動させています。

✧ 4.2a ジャイロセンサ

	PC	And	iOS
Firefox	×	○	○
Chrome	×	○	○

ジャイロセンサは、スマホの回転や、ユーザが向いている方位を測ります。パノラマ写真撮影や、空にかざして見るプラネタリウムアプリは、このセンサのおかげです。二つのサンプル（4.2a/4.2b）を通してジャイロデータの扱い方を学びます。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 4.2a 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。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 4.2a ¥ index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <meta name='viewport' content='width=device-width,initial-scale=1'>
    <title>4.2a ジャイロセンサ</title>
  </head>
  <body>
    <div id='msg'>ここにデータを表示</div>
    <script>
      // ジャイロセンサの値を得る deviceorientation イベント
      window.addEventListener('deviceorientation', getData);
      function getData(event) {
        // 回転角度 [degree]
        const alpha = event.alpha;    // 左右に傾ける回転
        const beta = event.beta;      // 前後に倒す回転
        const gamma = event.gamma;    // 左右に回る回転
        const msg = document.getElementById('msg'); // データを表示
        msg.innerHTML = 'a: ' + alpha + '<BR>' +
                        'b: ' + beta + '<BR>' +
                        'g: ' + gamma;
      }
    </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： **10.11.123.45/4.2a/**
- 画面の自動回転を OFF にしておくほうが確認しやすい



※1 Firefox でも動作しますが、授業で使うスマホとの相性のため Chrome を使います。



解説

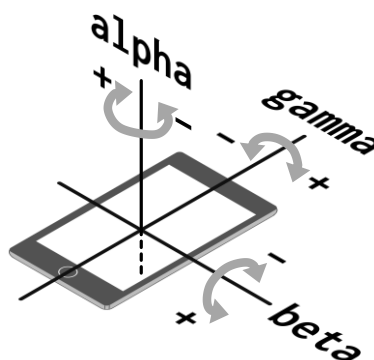
① `window.addEventListener('deviceorientation', getData);`

ジャイロセンサのデータが変化すると `deviceorientation` イベントが起きます。これが起きたらデータを取り出す関数（今回は `getData` と名付けた）を実行させます。

② `const alpha = event.alpha;`

ジャイロのデータは以下のような三種類が得られます。いずれも単位は度(degree)です。なお、ジャイロは Android も iOS も値の+/-は同じです。

- alpha : 画面を左右に傾ける向きの角度（時計回りが+、北が0度）
- beta : スマホを前後に倒す向きの角度（手前に起こす方向が+、水平が0度）
- gamma : スマホをひねる向きの角度（右ひねりが+、水平が0度）



✈ 4.2b ジャイロでパノラマ VR

	PC	And	iOS
Firefox	×	○	○
Chrome	×	○	○

ジャイロセンサの応用として、スマホを動かしてパノラマ写真の中を探検する、簡易なバーチャルリアリティ(VR)アプリを作ります。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 4.2b の中に、4.2a で作った index.html 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。
- CampusSquare から「PHC_04_素材」をダウンロードし、「4.2b 素材」の中にある 'norikura.jpg' (画像) を 4.2b 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 ¥ 4.2b ¥ index.html

(略)

```
<title>4.2b ジャイロでパノラマ VR</title>
</head>
<body>
  <div id='msg'>ここにデータを表示</div>
  <canvas width='300' height='400' id='canvas'></canvas>
  <script>
    // ジャイロセンサの値を得る deviceorientation イベント
    window.addEventListener('deviceorientation', getData);
    function getData(event) {
      // 回転角度 [degree]
      const alpha = event.alpha;    // 左右に傾ける回転
      const beta = event.beta;      // 前後に倒す回転
      const gamma = event.gamma;    // 左右に回る回転
      const msg = document.getElementById('msg'); // データを表示
      msg.innerHTML = 'a: ' + alpha + '<BR>' +
        'b: ' + beta + '<BR>' +
        'g: ' + gamma;
      moveImage(alpha);             // 写真を動かす (alpha のみ利用) ①
    }
    // パノラマ写真を読み込んで canvas に置く
    const canvas = document.getElementById('canvas');
    const context = canvas.getContext('2d');
    const img = new Image();        // Image オブジェクトを宣言
    img.src = 'norikura.jpg';       // パノラマ写真を指定
    img.addEventListener('load', drawImage); // 読み込めたら
    function drawImage() {          // canvas に写真を置く
      // 画像の L550,T0,W300,H400 を表示 (写真の中央付近)
      context.drawImage(img, 550, 0, 300, 400);
    }
  </script>
```

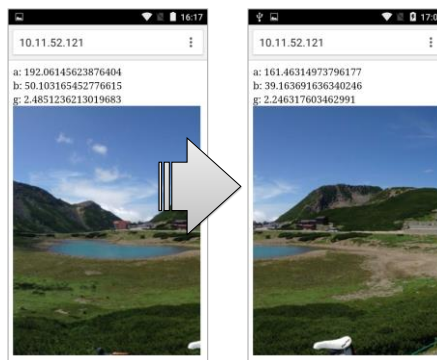
```
// 写真を左右に動かす
function moveImage(alpha) {
  context.clearRect(0, 0, canvas.width, canvas.height);
  const pan = alpha - 180; // 0~360度のalpha値を-180~180度に変換
  // 写真の表示位置を左右にずらす
  // 左右90度ずつで写真の端から端まで見えるように係数を6に調整
  context.drawImage(img, 550-6*pan, 0, 300, 400,
                    0, 0, 300, 400);
}
</script>
```

(以下略)



スマホで動作確認

- スマホの **Chrome** で確認※1
- アドレス例: 10.11.123.45/4.2b/

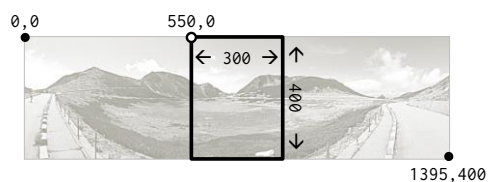


※1 Firefox でも動作しますが、授業で使うスマホとの相性のため Chrome を使います。

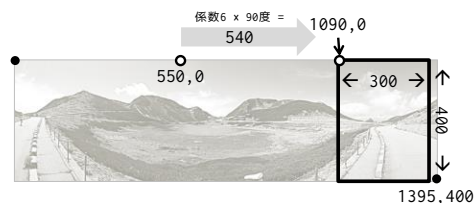
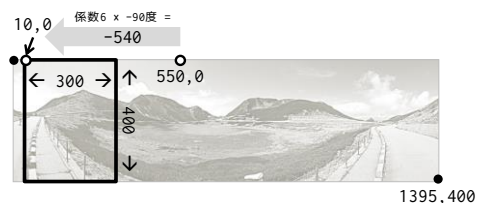


解説

alpha = 0 でパノラマの中央付近を表示させ、alpha に応じて表示位置を変更しています。



… alpha = 0 (北向きに立っている時)



… 左右に動かす

✧ 4.3a 照度センサ (Firefox 限)

	PC	And	iOS
Firefox	×	○	?
Chrome	×	×	×

照度センサは、明るさを測ります。周りの明るさに応じて画面の明るさが変わったり、裏向きにすると自動でスリープさせたりできるのはこのセンサのおかげです。二つのサンプル (4.3a/4.3b) を通して照度データの扱い方を学びます。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 4.3a 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。

※ ①や②などの丸数字は打ち込まないこと

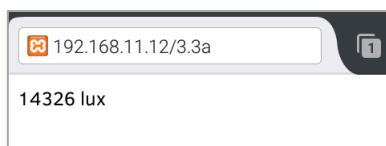
C: ¥ xampp ¥ htdocs ¥ 4.3a ¥ index.html

```
(略)
<title>4.3a 照度センサ (Firefox 限)</title>
</head>
<body>
  <div id='msg'>ここにデータを表示</div>
  <script>
    // 照度センサの値を得る devicelight イベント
    // Firefox のみ対応 (Chrome では動かない)
    window.addEventListener('devicelight', getData);
    function getData(event) {
      const lux = event.value; // 照度 [lux]
      const msg = document.getElementById('msg'); // データを表示
      msg.innerHTML = lux + ' lux';
    }
  </script>
</body>
</html>
```



スマホで動作確認

- スマホの  Firefox で確認
 - アドレス例: 10.11.123.45/4.3a/





解説

① `window.addEventListener('devicelight', getData);`

照度センサのデータが変化すると `devicelight` イベントが起きます。これが起きたらデータを取り出す関数（今回は `getData` と名付けた）を実行させます。

なお、`devicelight` イベントに対応しているブラウザは現状 Firefox のみです。

② `const lux = event.value;`

照度データは `value` が 1 個だけです。単位はルクスです。明るさの目安はおおむね下表のとおりです。ただし、スマホのセンサで得られるデータの精度は保証できません。

照度 (ルクス)	明るさの目安
100,000	晴天の昼の太陽光
65,000	晴天の午前 10 時の太陽光
32,000	曇天の昼の太陽光
10,000	曇天の日の出 1 時間後の太陽光
1,000	晴天時の日の入り 1 時間前の太陽光
600	デパートの売り場
100	街灯の下
10	口ウソクから 20cm
1	月明かり

✈ 4.3b 明るさチェッカー (Firefox 限)

	PC	And	iOS
Firefox	×	○	?
Chrome	×	×	×

照度センサの応用として、明るさを絵で示す簡易な明るさチェックアプリを作ります。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 4.3b の中に、4.3a で作った index.html 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。
- CampusSquare から「PHC_04_素材」をダウンロードし、「4.3b 素材」の中にある三つの画像ファイル ('moon.jpg', 'sunset.jpg', 'sunny.jpg') を 4.3b 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 ¥ 4.3b ¥ index.html

(略)

```

<title>4.3b 明るさチェッカー (Firefox 限)</title>
</head>
<body>
  <div id='msg'>ここにデータを表示</div>
  <img src='' id='brightness'>
  <script>
    // 照度センサの値を得る devicelight イベント
    // Firefox のみ対応 (Chrome では動かない)
    window.addEventListener('devicelight', getData);
    function getData(event) {
      const lux = event.value; // 照度 [lux]
      const msg = document.getElementById('msg'); // データを表示
      msg.innerHTML = lux + ' lux';
      showBrightness(lux); // 明るさを絵で示す
    }
    // 明るさを絵で示す
    const img = document.getElementById('brightness');
    const imgName = new Array('moon.jpg', 'sunset.jpg', 'sunny.jpg');
    function showBrightness(lux) {
      if(lux <= 100) { // 100lx 以下なら
        img.src = imgName[0]; // 月の絵
      } else if (lux > 100 && lux <=1000) { // 100<lux<=1000
        img.src = imgName[1]; // 夕暮れの絵
      } else { // それ以外 (lux>1000)
        img.src = imgName[2]; // 太陽の絵
      }
    }
  </script>

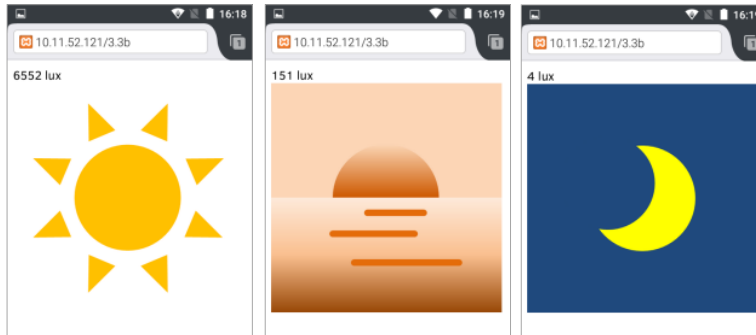
```

(以下略)



スマホで動作確認

- スマホの  **Firefox** で確認
 - アドレス例 : **10.11.123.45/4.3b/**



解説

- ① `const imgName = new Array('moon.jpg', 'sunset.jpg', 'sunny.jpg');`
 明るさがイメージとして分かるように、月の絵、夕日の絵、さんさんと輝く太陽の絵、三種類を用意しました。
- ② `function showBrightness(lux) { ... }`
 明るさが 100 ルクス以下なら月の絵、100 ルクスより大きく 1,000 ルクス以下なら夕日の絵、1,000 ルクスより大きいなら太陽の絵を、img 要素に表示しています。
- ③ `if(...) { ... } else if (...) { ... } else { ... }`
 条件分岐が三つ以上になる場合は、`else if () { ... }` を使います。これでいくつでも条件を増やすことができます。
- ④ `lux > 100 && lux <=1000`
`&&` は「かつ」という論理演算子です。`&&` の左と右が同時に成り立つとき真(true)になります※。この場合、lux が 100 より大きい「かつ」lux が 1000 以下、という条件を示しています。

※ `&&` は論理積 (AND) と呼びます。高校で「 \cap 」や「 \wedge 」という記号があったかと思います。

実は「2.6 My カメラアプリ」のコード内には `||` という論理演算子がありました。「または」という意味です。

`||` の左右のどちらかが成り立てば真(true)になります。これは論理和 (OR) と呼び、高校で「 \cup 」や「 \vee 」という記号があったかと思います。

✧ 4.4a 近接センサ (Firefox 限)

	PC	And	iOS
Firefox	×	○	?
Chrome	×	×	×

近接センサは、ユーザとの距離を測ります（たいていのスマホは近い／遠い程度ですが）。スマホを耳にあてて通話する時に画面がロックされるのはこのセンサのおかげです。二つのサンプル（4.4a/4.4b）を通して近接データの扱い方を学びます。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 4.4a 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。


※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 4.4a ¥ index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <meta name='viewport' content='width=device-width,initial-scale=1'>
    <title>4.4a 近接センサ (Firefox 限)</title>
  </head>
  <body>
    <div id='msg'>ここにデータを表示</div>
    <script>
      // 近接センサの値を得る deviceproximity イベント
      // Firefox のみ対応 (Chrome では動かない)
      window.addEventListener('deviceproximity', getData);
      function getData(event) {
        const prox = event.value;           // 距離 [近い=0 / 遠い=1]
        const msg = document.getElementById('msg'); // データ表示
        msg.innerHTML = prox;
      }
    </script>
  </body>
</html>
```



スマホで動作確認

- スマホの  Firefox で確認
 - アドレス例: 10.11.123.45/4.4a/
 - 通話用スピーカー付近に手などを近づけると 0 が、遠ざけると 1 が表示されます



解説

① `window.addEventListener('deviceproximity', getData);`

照度センサのデータが変化すると `deviceproximity` イベントが起きます。これが起きたらデータを取り出す関数（今回は `getData` と名付けた）を実行させます。

なお、`deviceproximity` イベントに対応しているブラウザは現状 Firefox のみです。

② `const prox = event.value;`

近接データは `value` が 1 個だけです。多くのスマホでは、近い=0 / 遠い=1 です。中には cm 単位でデータが得られるスマホもあるようです。



コラム③ イベント設定の三つの方法

授業ではイベント設定に `xxx.addEventListener(event, function);` という書き方を使っています。実は他にも 2 種類、違う書き方があります。以下の①と②の書き方は授業では使わない予定ですが、世の中のサンプルを読むときの参考にしてください。

授業の書き方： `btn.addEventListener('click', showMsg);`

別の書き方①： `btn.onclick = showMsg;`

別の書き方②： `<input type='button' id='btn' onclick='showMsg'>`

①は、`btn` オブジェクトの `onclick` 属性に `showMsg` 関数を指定する方法です。

②は、`input` タグの `onclick` 属性に `showMsg` 関数を指定する方法です。

どの書き方でも同じ動作をします。どれが良い／悪いということはありませんが…

①の書き方は、ひとつのイベントに複数の関数を設定できません。例えば、ボタンを `click` した時、`showMsg` に加えて、別の関数 `playSound` も実行したいとします。`addEventListener` の場合、もう一行 `btn.addEventListener('click', playSound);` と書くことで、`click` により二つの関数を実行できます。まさに `add` する（加える）ことができるのです。①の場合、もし `btn.onclick = playSound;` と書くと、最初にした `btn.onclick = showMsg` は消え、`showMsg` は実行できなくなります。

②の書き方は、近ごろは避けられる傾向です。HTML はコンテンツを記述するもの、CSS はスタイル（見た目）を記述するもの、JavaScript はダイナミックな動作を記述するもの、というように分けて考える・書く方式が世の流れです。分けて書くと、`html` ファイル全体が見やすくなることと、大きな仕事の場合には分業できるメリットがあるからです。②の場合、HTML の中に JavaScript が入り込んでいます。HTML（コンテンツ）を書き換えた時に、JavaScript（動作）にも影響が出てしまう危険性があります。

そのような理由でこの授業では、最も汎用性が高い `addEventListener` を使っています。

✈ 4.4b 盗難防止アラーム (Firefox 限)

	PC	And	iOS
Firefox	×	○	?
Chrome	×	×	×

近接センサの応用として、誰かがスマホに手を伸ばすと警告するアプリを作ります。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 4.4b の中に、4.4a で作った index.html 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。
- CampusSquare から「PHC_04_素材」をダウンロードし、「4.4b 素材」の中にある 'tasukete.mp3' (音声) と 'tasukete.png' (画像) を 4.4b 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 ¥ 4.4b ¥ index.html

(略)

```

<title>4.4b 盗難防止アラーム (Firefox 限)</title>
<style>
  #img {visibility: hidden;} ①
</style>
</head>
<body>
  <div id='msg'>ここにデータを表示</div>
  <audio src='tasukete.mp3' id='snd'></audio>
  <img src='tasukete.png' id='img'>
  <script>
    window.addEventListener('deviceproximity', getData);
    function getData(event) {
      const prox = event.value;           // 距離 [近い=0 / 遠い=1]
      const msg = document.getElementById('msg'); // データ表示
      msg.innerHTML = prox;
      alarm(prox); // アラームを鳴らす
    }

    // アラームを鳴らす
    const snd = document.getElementById('snd');
    const img = document.getElementById('img');
    function alarm(prox) {
      if(prox == 0) { // 距離が0 (近い) なら
        snd.play(); // 音声を再生 ②
        img.style.visibility = 'visible'; // 画像を表示 ③
        window.setTimeout(hideImg, 3500); // 一定時間後に画像隠す ④
      }
    }
  </script>

```

```
// 画像を隠す
function hideImg() {
    img.style.visibility = 'hidden';
}
</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： **10.11.123.45/4.4b/**
 - 通話用スピーカーのあたりに手などを近づけると…



紙面の都合で横倒しに表示しています。



解説

① **#img {visibility: hidden;}**

id が img の要素(ここでは img タグ)を非表示にする(隠す)CSS です。visibility 属性に hidden を指定すると非表示、visible を指定すると表示されます。最初は画像を隠しておくために hidden を指定しています。

② **snd.play();**

audio タグに読み込んでおいた音声ファイルを再生する play() です。一時停止は pause()、止めるのは stop() です。

③ **img.style.visibility = 'visible';**

img 要素のスタイルの visibility 属性を visible にして、画像を表示しています。

④ **window.setTimeout(hideImg, 3500);**

指定時間後に関数を実行させる setTimeout(関数, 時間) です。時間はミリ秒で指定します。ここでは 3.5 秒後に hideImg を 1回 実行して、画像を非表示に戻しています。

なお、第2回の時計アプリで使った setInterval(関数, 時間) は、指定した時間間隔で 繰り返し 関数を実行します。

✈ 4.5a GPS センサ

	PC	And	iOS
Firefox	x	○	○
Chrome	x	○	○

GPS センサは、人工衛星からの電波をもとにユーザがいる位置を測ります。地図アプリや道案内（カーナビ）アプリ、天気アプリで自分の居場所が分かるのはこのセンサのおかげです。二つのサンプル（4.5a/4.5b）を通して GPS データの扱い方を学びます。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 4.5a 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。


※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 4.5a ¥ index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8'>
    <meta name='viewport' content='width=device-width,initial-scale=1'>
    <title>4.5a GPS センサ</title>
  </head>
  <body>
    <div id='msg'>ここにデータを表示</div>
    <script>
      // 現在位置が変わる度に関数を呼ぶ geolocation.watchPosition()
      navigator.geolocation.watchPosition(getData); ①
      // 現在位置が変わった時に実行される関数
      function getData(pos){
        // 緯度・経度 [degree] を取得
        const lat = pos.coords.latitude; // 緯度 ②
        const lng = pos.coords.longitude; // 経度
        const msg = document.getElementById('msg'); // データ表示
        msg.innerHTML = '緯度: ' + lat + '<BR>' +
          '経度: ' + lng;
      }
    </script>
  </body>
</html>
```



スマホで動作確認

- スマホの  **Firefox** で確認[※]
 - アドレス例： **10.11.123.45/4.5a/**
 - 屋内では人工衛星の電波が届かないので、空が見える窓際か、少し外に出て試しましょう

10.11.52.121	⋮
緯度: 35.1571538	
経度: 137.0329495	

※1 Chrome でも問題なく動作しますが、流れ上、Firefox を使っています。



解説

- ① **navigator.geolocation.watchPosition(getData);**
 現在位置のデータが変わる度に（繰り返し）関数を呼ぶ `geolocation.watchPosition()` です。
 他に、1 回だけ現在位置を取得する `geolocation.getCurrentPosition()` もあります。
- ② **const lat = pos.coords.latitude;**
 緯度を得る `coords.latitude` です。その下の `coords.longitude` は経度です。単位はいずれも度 [degree] です。



コラム⑭ GPS と geolocation

GPS(Global Positioning System)は、人工衛星が発している電波を受けて位置を測定（測位）する、アメリカの軍用システムです。他に同様のものとして、ロシアの GLONASS、EU の Galileo、中国の北斗、日本の「みちびき」などがあります（「みちびき」はアメリカの GPS の補完システムです）。一般的にはこれらのシステムをまとめて GPS と呼ぶことが多いです。

JavaScript の geolocation は、これら GPS のデータを取得しています。さらに geolocation は GPS だけでなく、Wi-Fi の電波なども使って、屋内や建物の陰などの GPS 電波が届かない場所でも、それなりの位置データを取得できます。そのため、GPS を搭載していないタブレットなどでも、Wi-Fi に接続していればある程度の位置が分かります。

なお、GPS データには 10～20m の誤差はあるものと思ひましょう。条件が悪いと 50～100m の誤差が出ることもあります。過度な期待は禁物です。ただし、「みちびき」が本格運用されると日本国内では数 cm の誤差になると言われています（2019 年ごろを予定）。

✈ 4.5b ここにいるよ

	PC	And	iOS
Firefox	x	○	○
Chrome	x	○	○

GPS センサの応用として、現在位置を Google マップに示すアプリを作ります。



HTML と JavaScript を入力

- C: ¥ xampp ¥ htdocs ¥ 4.5b の中に、4.5a で作った index.html 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。
- CampusSquare から「PHC_04_素材」をダウンロードし、「4.5b 素材」の中にある 'gmap_api.txt' に書かれた 1 行をコピーしておきましょう。

※ ①や②などの丸数字は打ち込まないこと

C: ¥ xampp ¥ htdocs ¥ 4.5b ¥ index.html

(略)

```
<title>4.5b ここにいるよ</title>
```

```
<style>
```

```
  #map {width: 300px; height: 400px;}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
  <div id='msg'>ここにデータを表示</div>
```

```
  <div id='map'></div>
```

```
  <!-- Google マップを使うための外部 script(API)の呼び出し -->
```

```
  <!-- この 1 行は 4.5b 素材 内の gmap_api.txt からコピー -->
```

```
  <script src='https://maps.googleapis.com/maps/api/js?
```

```
    key=[ ]'></script>
```

①

```
<script>
```

```
  navigator.geolocation.watchPosition(getData);
```

```
  function getData(pos){
```

```
    const lat = pos.coords.latitude; // 緯度
```

```
    const lng = pos.coords.longitude; // 経度
```

```
    const msg = document.getElementById('msg'); // データ表示
```

```
    msg.innerHTML = '緯度: ' + lat + '<BR>' +
```

```
                  '経度: ' + lng;
```

```
    showOnMap(lat, lng); // 現在位置を Google マップで表示
```

```
  }
```

```
  // Google マップを表示 (ひとまず仮の場所)
```

```
  const mapDiv = document.getElementById('map'); // 地図を置く div
```

```
  const iniPos = new google.maps.LatLng(35.1709, 136.8815); // 初期位置: 名駅 ②
```

```
  const mapOpt = {center: iniPos, zoom: 16}; // 地図オプション ③
```

```
  const gmap = new google.maps.Map(mapDiv, mapOpt); // 地図を表示 ④
```

```
  // 地図にマーカを表示
```

```
  const markOpt = {map: gmap, position: iniPos}; // マーカオプション ⑤
```

```
  const mark = new google.maps.Marker(markOpt); // マーカを表示 ⑥
```



```
// センサで得られた位置に地図とマーカを移動
function showOnMap(lat, lng) {
  const myPos = new google.maps.LatLng(lat, lng); // 現在位置 ⑦
  gmap.setCenter(myPos); // 地図の中心を移動 ⑧
  mark.setPosition(myPos); // マーカを移動 ⑨
}
</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： **10.11.123.45/4.5b/**
 - 屋内では人工衛星の電波が届かないので、空が見える窓際か、少し外に出て試しましょう



紙面の都合で横向きに表示しています。

※1 Chrome でも問題なく動作しますが、流れ上、Firefox を使っています。




解説

- ① **<script src='https://maps.googleapis.com/maps/api/js?key=xxx...'></script>**
 Google マップを自分のアプリで使うための 1 行です。Google のスクリプトを引用しています。
 この際、key= に続けて「API キー」という文字列 (パスワードのようなもの) を書きます。この授業ではキャンパススクエアからダウンロードできるようにした gmap_api.txt 内に key を書いてありますから、それをコピーして使ってください。授業以外でも Google マップを用いたアプリを開発したい場合は、付録 6 を参考に、自身で API キーを取得してください。
- ② **const iniPos = new google.maps.LatLng(35.1709, 136.8815);**
 緯度と経度の情報を入れておく、Google マップの LatLng オブジェクトです。スマホが GPS で現在位置を測るのに時間がかかることがあるため、ここではとりあえずの表示位置として名古屋駅付近の緯度・経度を指定しています。
- ③ **const mapOpt = {center: iniPos, zoom: 16};**
 地図の表示オプションの指定です。表示位置②と、地図の表示倍率を指定します。


④ **`const gmap = new google.maps.Map(mapDiv, mapOpt);`**

Google マップを実際に表示させている命令です。地図の置き場にする HTML 要素と、地図の表示オプション③を指定します。

⑤ **`const markOpt = {map: gmap, position: iniPos};`**

マーカ  の表示オプションの指定です。マーカを置く地図④と、表示位置②を指定します。

⑥ **`const mark = new google.maps.Marker(markOpt);`**

マーカ  を表示する命令です。マーカの表示オプション⑤を指定します。


⑦ **`const myPos = new google.maps.LatLng(lat, lng);`**

GPS センサで測定できたら、LatLng オブジェクトに自分がいる場所の緯度・経度を入れます。

⑧ **`gmap.setCenter(myPos);`**

表示する地図の中心を、自分がいる場所⑦に設定します。

⑨ **`mark.setPosition(myPos);`**

マーカ  を自分がいる場所⑦に設定します。



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 (11 個)
 - － 4.1a/b/c 4.2a/b 4.3a/b 4.4a/b 4.5a/b



本日の課題

- ① 4.1c、4.2b、4.5a 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 3 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 04」から提出

【提出締切】 **10月27日(木) 15:00** (遅れてしまった場合は担当教員に相談のこと)