

今回の内容

目標

様々な場所に人魚を配置して、その場所に行くと人魚をゲットできるゲーム「マーメイドGO」を作ります。



今回の目標

使うもの

- Windows のパソコン (Windows 7 以上)
- Visual Studio Code ([コチラを参考にインストール](#)。好みのエディタで構いません)
- パソコンのサーバ機能 XAMPP (Apache) ([こちらを参考に設定](#))
- スマホまたはタブレット (なんでも構いません)
- 無線 LAN (Wi-Fi) 環境、またはスマホのテザリング機能 ([こちらを参考に設定](#))

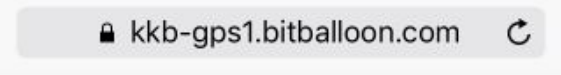
ステップ

1. [GPS の値を得る](#)
2. [Google マップに位置を表示](#)
3. [マーメイドGO](#)

▲TOP

1. GPS の値を得る

なにはともあれ、GPS の値 (位置情報) を取得して、そのまま値を表示してみましょう。



緯度, 経度: 35.15733467073155,
137.03382551716197
精度: 65

今から作るもの

プログラミング

Visual Studio Code（または好みのエディタ）を立ち上げて、以下のコードを入力しましょう。[こちらで作ったテンプレート](#)をもとに書き始めると効率的です。ファイルは、C:\xampp\htdocs フォルダの中に「gps1」というフォルダを作って、その中に「index.html」として保存してください。それぞれのコードが何をしているのか考えながら入力していきましょう。

```

1. <!DOCTYPE html>
2. <html lang="ja">
3. <head>
4. <meta charset="utf-8">
5. <meta name="viewport" content="width=device-width,initial-scale=1">
6. <title>GPS の値を得る</title>
7. </head>
8.
9. <body>
10. <div id="txt">ここにデータを表示</div>          <!-- データを表示するdiv要素 -->
11.
12. <script>
13. // GPS センサの値が変化したら何らか実行する geolocation.watchPosition メソッド
14. navigator.geolocation.watchPosition( (position) => {
15.   var lat = position.coords.latitude;           // 緯度を取得
16.   var lng = position.coords.longitude;          // 経度を取得
17.   var accu = position.coords.accuracy;          // 緯度・経度の精度を取得
18.   displayData(lat, lng, accu);                  // displayData 関数を実行
19. }, (error) => {                                // エラー処理（今回は特に何もしない）
20. }, {
21.   enableHighAccuracy: true                      // 高精度で測定するオプション
22. });
23.
24. // データを表示する displayData 関数
25. function displayData(lat, lng, accu) {

```

動作確認

GPS のデータ（位置情報）を使った Web アプリは、動作確認にちょっとコツが必要です（理由は後ほど補足します）。Android スマホの場合は、Firefox ブラウザをインストールしてから[次に進んでください](#)。iOS の場合、またはAndroid だけど Firefox はインストールしたくない場合は[こちらに進んでください](#)。

Android スマホの Firefox の場合

XAMPP Control Panel で Apache を Start させてください（不明な場合は[こちらを参考](#)にしてください）。

プログラムを書いているパソコンと、動作確認するスマホが、同じ LAN につながっていることを確認します（不明な場合は[こちらを参考](#)にしてください）。

Windows の「コマンド プロンプト」で ipconfig して、パソコンの IP アドレスを調べてください（不明な場合は[こちらを参考](#)にしてください）。

スマホのブラウザ（Safari や Chrome）を開き、アドレス欄に以下のように入力します。以下の「10.11.52.81」の部分は上で調べた IP アドレスです。

- 10.11.52.81/gps1

[こちらに進んで](#)ください。

iOS の場合

もし外部にホームページスペースを持っていて、かつそこが https 接続に対応していたら、そこに ftp で gps1 フォルダを配置して、スマホでアクセスください。この時、**アドレスの先頭に「https://」**と入力してください。

- **https://**自分のHPスペースのアドレス/**gps1**

自分のホームページスペースを持っていない人、および、持っていない人も https 接続に対応していなければ、以下のようにしてください。

パソコンで [Netlify Drop](#) などの無料ホスティングサービスにアクセスします。例えば Netlify Drop であれば、そのトップページに「gps1」フォルダをドラッグ&ドロップしてください。



Netlify Drop にアップロード

アップロードされたら、表示されたアドレスをスマホのブラウザで開きます。この時、**アドレスの先頭に「https://」**と入力してください。

この手順についてよく分からない場合は[こちらを参考](#)にしてください。

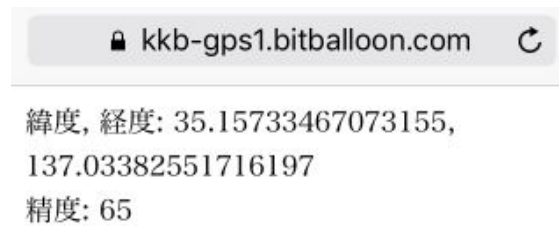
ここから共通の手順

ブラウザで以下のような確認のダイアログが出たら「OK」をタップして位置情報の利用を許可してください。もし、スマホの位置情報サービス自体を OFF にしている場合は、スマホの設定画面で ON にして、使っているブラウザで位置情報サービスが使えるように設定してください。



位置情報を利用するかどうかの確認

スマホで以下のように表示されます。



アクセスすると

緯度と経度の値がうまく表示されない場合は、スマホを持ってベランダや庭先など、空が見える場所に出てみましょう。

表示された緯度と経度の単位は「度」です。緯度の値（0～90度）は赤道が 0度で、プラスなら北緯、マイナスなら南緯です。経度の値（0～180度）は子午線が 0度で、プラスなら東経、マイナスなら西経です。日本であれば、緯度も経度もプラスの値です。

と、数字だけ見ても正しい値かどうか分かりませんね。確かめてみましょう。まずは以下のように緯度と経度の値の部分を選択してコピーしてください。



緯度と経度の値をコピー

コピーした値を、ブラウザのアドレス欄に貼り付けて開いてみてください。スマホで文字列の範囲選択コピーが難しい場合は、緯度と経度の値を小数第4位くらいまでメモして、「緯度の値,経度の値」のように値をカンマで区切ってアドレス欄に入力してみてください。すると、Google マップが表示されて、自分がいる場所が表示されます。



GPS データの確認

解説

プログラムのポイントは、14～22行目の **geolocation.watchPosition** メソッドです。**watchPosition**(測位に成功した時の関数, 測位に失敗した時の関数, {測位オプション}) とすると、GPS の値が変化するたびに、成功した時の関数が呼ばれます。この関数の中で、**coords.latitude** (緯度) と **coords.longitude** (経度) という属性を参照しています。

```

13. // GPS センサの値が変化したら何らか実行する geolocation.watchPosition メソッド
14. navigator.geolocation.watchPosition( (position) => {
15.   var lat = position.coords.latitude;      // 緯度を取得
16.   var lng = position.coords.longitude;     // 経度を取得
17.   var accu = position.coords.accuracy;     // 緯度・経度の精度を取得
18.   displayData(lat, lng, accu);             // displayData 関数を実行
19. }, (error) => {                           // エラー処理 (今回は特に何もしない)
20. }, {
21.   enableHighAccuracy: true                 // 高精度で測定するオプション
22. });

```

coords.accuracy は位置情報の精度 (誤差) で、単位は m です。この値くらいは緯度と経度の値がずれている可能性があるという値です。

なお、今回は第2引数の「測位に失敗した時の関数」は中身は空っぽにしてあります。また、第3引数のオプションは { } で囲んで指定します。今回は「enableHighAccuracy: true」というオプションを指定して、できるだけ高精度で測位させています。GPS 電波が届いている場合は GPS を使い、GPS 電波が届いていない場合は携帯電話基地局や Wi-Fi の電波などから推定した位置の値を取得します。ちなみにここのオプションには他にもいろいろ設定できます。自分で調べてみましょう。

参考

完成品を[こちら \(大学\)](#) または[こちら \(Netlify Drop\)](#) に置いてありますからスマホで開いてみてください。

補足

GPS などを使ってユーザの居場所が分かる geolocation は、もし悪用されるとプライバシーが侵害される可能性があります。あるホームページにアクセスしたらユーザの居場所が分かるわけですから、裏技を使えばいろいろ個人情報を盗み出す方法もあり得ます。そのため多くのブラウザでは、https 接続（http より安全な接続）が可能なサイトでないと geolocation を使って位置情報を測れないようにしています。このサイトの記事では XAMPP（Apache）を自分のパソコンにインストールして http 接続でアプリのテストをしています。Apache で https 接続を可能にする方法もありますが、個人レベルでは簡単ではありません。そこで、もっとも簡単に https 接続が可能な環境を用意してテストするために、Netlify Drop などのホスティングサービスを利用する方法を紹介しています。なお、この記事を書いている時点で Android 版の Firefox だけは https 接続でなくても geolocation で現在位置を測れますが、おそらく近いうちに測れなくなると考えられます。

[▲TOP](#)

2. Google マップに位置を表示

位置（緯度と経度）が測れましたが、いちいち値をコピーして Google マップで表示させるのは面倒です。Google マップそのものを自分のアプリの中に組み込んで、自分がいる位置を表示できるアプリを作ります。



プログラミング

Visual Studio Code（または好みのエディタ）を立ち上げて、さきほどの gps1 の index.html をもとに以下のコードを入力しましょう。**<title> タグ ~ </script> タグ の部分のみを掲載します（このままコピーするだけでは動きません）**。ファイルは、C:\xampp\htdocs フォルダの中に「gps2」というフォルダを作って、その中に「index.html」として保存してください。それぞれのコードが何をしているのか考えながら入力しましょう。

1. `<title>Google マップに位置を表示</title>`
2. `<style>`
3. `/* Google マップを表示させるためには style 内で width と height の指定が必要 */`
4. `#mapDiv {width: 100%; height: 400px;}`

```

5. </style>
6. </head>
7.
8. <body>
9. <div id="txt">ここにデータを表示</div>          <!-- データを表示するdiv要素 -->
10. <div id="mapDiv"></div>                          <!-- 地図を置くdiv要素 -->
11.
12. <script>
13. var mapDiv = document.getElementById("mapDiv");    // 地図を置く場所
14. var gmap;                                         // Googleマップの Map オブジェクトのための変数
15. var mark;                                         // Googleマップの Marker オブジェクトのための変数
16.
17. // GPS センサの値が変化したら何らか実行する geolocation.watchPosition メソッド
18. navigator.geolocation.watchPosition( (position) => {
19.   var lat = position.coords.latitude;             // 緯度を取得
20.   var lng = position.coords.longitude;            // 経度を取得
21.   var accu = position.coords.accuracy;            // 緯度・経度の精度を取得
22.   displayData(lat, lng, accu);                    // displayData 関数を実行
23.   showMyPos(lat, lng);                            // showMyPos 関数を実行
24. }, (error) => {                                   // エラー処理（今回は特に何もしない）
25. }, {

```

動作確認

このアプリを試すには、Google のとあるページで、自分のアプリの中で Google マップを使うための登録をして、「API キー」という文字列を入手する必要があります（無償です）。

パソコンで[こちらにアクセス](#)してください。



右上にある「キーを取得」をクリックします。すると以下のようなダイアログが開くので「続ける」をクリックします。



API アクティベートの手順

すると Google アカウントのログイン画面になります。Google アカウントを既に持っていればログインしてください。まだ Google のアカウントを持っていないければ、下のほうにある「アカウントを作成」をクリックして、アカウントを作ってください（無償です）。



アカウント1つですべての Google サービスを。

Google API Console に移動するにはログイン



アカウントを作成

Google アカウントでログイン

ログインすると次のような画面になります。「プロジェクトを作成」を選択して「続行」ボタンをクリックしてください。

Google API コンソール での Google Maps JavaScript API Google Maps Geocoding API Google Maps Directions API Google Maps Distance Matrix API Google Maps Elevation API Google Places API Web Service のアプリケーションの登録

Google API コンソールでは、アプリケーションの管理と API 使用のモニタリングができます。

アプリケーションを登録するプロジェクトの選択

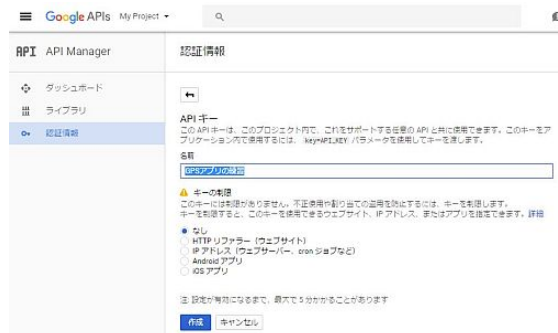
1つのプロジェクトですべてのアプリケーションを管理できます。また、アプリケーションごとに異なるプロジェクトを作成することもできます。

プロジェクトを作成

続行

プロジェクトを作成

少し待つと以下のような画面になります。「名前」欄に分かりやすい名前を入力します。例えば「GPSアプリの練習」のように。その後「作成」をクリックします。



API キーの作成

すると「API キーを作成しました」という画面になり、「自分の API キー」という欄に文字列（意味不明で長い）が表示されます。この「API キー」という文字列を自分のアプリの中に書くことで、自分のアプリで Google マップが使えるようになります。この文字列をコピーしてください。



API キーが作成された

gps2 の index.html の以下の部分を見てください。

```
65. <!-- Google マップを読み込むための外部スクリプトの読み込み -->
66. <!-- API_KEY の部分を自身で取得した APIキーで置き換える -->
67. <!-- callback=initMap により、マップの準備ができたなら initMap 関数が実行される -->
68. <script src="https://maps.googleapis.com/maps/api/js?key=API_KEY&callback=initMap">
69. </script>
```

68行目の <script> タグの src に "https://maps.googleapis.com/maps/api/js?key=API_KEY&callback=initMap" と指定しています。この「API_KEY」の部分を、上で作った「API キー」の文字列で置き換えてください。これで自分のアプリの中で Google マップを使う準備ができました。

[ステップ1と同じやり方](#)で動作確認してください。

Android の Firefox が使えるなら、アドレス欄に以下のように入力します。以下の「10.11.52.81」の部分はパソコンで ipconfig して調べた IP アドレスです。

- 10.11.52.81/gps2

自分のホームページスペースを持っていて、かつそこが https 接続に対応していれば...

- https://自分のHPスペースのアドレス/gps2

自分のホームページスペースを持っていない人、および、持っていない人も https 接続に対応していなければ、パソコンで [Netlify Drop](#) などの無料ホスティングサービスに「gps2」フォルダをアップして、表示されたアドレスをスマホのブラウザで開きます。この時、**アドレスの先頭に「https://」と入力してください。**

スマホで以下のように表示されます。



自分のいる場所が Googleマップの上に表示されていれば成功です。できれば少し歩き回ってみて、自分のいる場所が動くことも確認しましょう。

この時、**危険ですから、自動車や自転車などで動き回ったり、動きながらスマホを見るのは絶対にやめましょ**う。

解説

gps1 から加えられたのは 37行目以降です。showMyPos() 関数、initMap() 関数、それから 68・69行目の <script>タグです。

68・69行目の <script>タグの中身が、自分のアプリの中で Google マップを使う設定です。このアプリにアクセスすると、"https://maps.googleapis.com/maps/api/js" というアドレスに自動でアクセスします。このアドレスが、アプリに組み込むための Google マップのアドレスです。このマップにアクセスするためには「API キー」が必要です。そこで「key=API_KEY（さきほど取得したAPI キーを指定）」とすることで API キーを指定しています。さらに「callback=initMap」と指定することで、マップの読み込みが終わったら initMap() 関数を呼ぶようにしています。

```
65. <!-- Google マップを読み込むための外部スクリプトの読み込み -->
66. <!-- API_KEY の部分を自身で取得した APIキーで置き換える -->
67. <!-- callback=initMap により、マップの準備ができれば initMap 関数が実行される -->
68. <script src="https://maps.googleapis.com/maps/api/js?key=API_KEY&callback=initMap">
69. </script>
```

initMap() 関数では、geolocation.getCurrentPosition メソッドを実行しています。さきほども使った watchPosition メソッドは位置の値が変わるたびに繰り返し実行されますが、getCurrentPosition は「1回だけ」位置の値を取得します。() 内は watchPosition と同じで、(測位に成功した時の関数, 測位に失敗した時の関数, {測位オプション}) です。

```
43. // 地図の初期化
44. function initMap() {
45.     // 1回だけ現在位置を測定する getCurrentPosition メソッド
46.     navigator.geolocation.getCurrentPosition( (position) => {
47.         var lat = position.coords.latitude;    // 緯度を取得
48.         var lng = position.coords.longitude;    // 経度を取得
49.         var initPos = new google.maps.LatLng(lat, lng); // 初期位置を指定
50.         gmap = new google.maps.Map(mapDiv, {    // Map オブジェクトを作成して mapDiv に表示
51.             center: initPos,                    // 地図の中心を initPos に設定
52.             zoom: 16                            // ズーム倍率
53.         });
54.         mark = new google.maps.Marker({        // Marker オブジェクトを作成
55.             map: gmap,                          // gmap の上に表示する
56.             position: initPos,                  // initPos の位置に
57.         });
58.     }, (error) => {                            // エラー処理（今回は特に何もしない）
59.     }, {
60.         enableHighAccuracy: true                // 高精度で測定するオプション
61.     });
62. }
```

測位に成功した時の関数内で、緯度と経度を取得した後、Google マップを表示して、マーカーを表示しています。まず Google マップの LatLng というオブジェクト initPos を作って、測った緯度と経度の値を入れます。次に Map オブジェクト（これが Google マップそのもの）を作っています。Map() の () 内は (地図を表示する html 要素, {表示オプション}) です。表示オプションの { } 内で、「center（地図の中心の場所）」に測った緯度と経度、「zoom（地図の表示倍率）」に適当に 16倍を指定しています。

次に Marker オブジェクトを作っています。よく見る Google マップの赤いピンの部品です。Marker({ }) の { } 内でマーカーの表示オプションを指定します。ここでは、「map（マーカーを置く地図）」に上で作った Map オブジェクト (gmap) を、「position（マーカーを置く場所）」に測った緯度と経度を指定しています。

地図の準備が終わったら、watchPosition が成功した時の関数内で、showMyPos() 関数を呼んでいます。

```
23. showMyPos(lat, lng);
```

```
// showMyPos 関数を実行
```

showMyPos() 関数で自分のいる場所を表示しています。さきほどと同じように、LatLng オブジェクト (myPos) を作って、測った緯度と経度の値を入れています。さらに、initMap() 関数の中で作った Map オブジェクト gmap の setCenter メソッドで、自分のいる場所を地図の中心にしています。同じように、initMap() 関数の中で作った Marker オブジェクト mark の setPosition メソッドで、自分のいる場所にマーカーを表示しています。

```
36. // 自分の位置を表示する showMyPos 関数
37. function showMyPos(lat, lng) {
38.     var myPos = new google.maps.LatLng(lat, lng); // Googleマップの LatLng オブジェクトを作成
39.     gmap.setCenter(myPos); // gmap の中心を myPos の位置にする
40.     mark.setPosition(myPos); // mark の位置を myPos にする
41. }
```

さあ、これで、いつも便利に使っている Google マップを自分のアプリに組み込むことができました。

参考

完成品を[こちら \(大学\)](#) または[こちら \(Netlify Drop\)](#) に置いてありますからスマホで開いてみてください。

補足

Google マップの API は無償で使うことができますが、無償の場合は制限があります。[こちらのページの「Usage Limits for JavaScript API Services」](#)に書いてあるとおり、1日に 2,500回のリクエストに制限されます。とはいえ、個人で試すぶんには全く問題ないと思います。しかし、あなたの作ったアプリがものすごく楽しくて、とても多くのアクセスがある場合は、2,500回を超えるかもしれません。ちなみに、使えるリクエスト回数は、1,000回あたり 0.5 アメリカドルで、上限 100,000回のリクエストまで追加することができます。

▲TOP

3. マーメイドGO

Google マップを自分のアプリに組み込むことができましたが、これだけでは、いつものように Google マップの純正アプリそのものを使えばいいだけです。自分のアプリの中に組み込むと、もっといろいろなことができます。ここでは、マップの様々な場所に人魚を配置して、その場所に行くと人魚をゲットできるゲーム「マーメイドGO」を作ります。「ポケモンGO」には遠く及びませんが、それっぽいものを自分で作ってみましょう。



今から作るもの

プログラミング

Visual Studio Code（または好みのエディタ）を立ち上げて、さきほどの gps2 の index.html をもとに以下のコードを入力しましょう。<title> タグ ~ </script> タグ の部分のみを掲載します（このままコピペするだけでは動きません）。ファイルは、C:\xampp\htdocs フォルダの中に「gps3」というフォルダを作って、「index.html」として保存してください。それぞれのコードが何をしているのか考えながら入力していきましょう。gps2 から加えているのは★印の部分です。

```

1. <title>マーメイド GO</title>
2. <style>
3.   /* Google マップを表示させるためには style 内で width と height の指定が必要 */
4.   #mapDiv {width: 100%; height: 400px;}
5. </style>
6. </head>
7.
8. <body>
9.   <div id="mapDiv"></div>          <!-- 地図を置くdiv要素 -->
10.  <img id="getImg" src="" hidden>    <!-- ★捕獲した人魚を表示するimg要素 -->
11.  <canvas id="cap" width="300" height="60"></canvas> <!-- ★捕獲済人魚を入れるcanvas要素 -->
12.
13. <script>
14.   var mapDiv = document.getElementById("mapDiv"); // 地図を置く場所
15.   var gmap; // Googleマップの Map オブジェクトのための変数
16.   var mark; // Googleマップの Marker オブジェクトのための変数
17.
18.   var mermaids = []; // ★人魚の情報を入れる変数
19.   var captured = []; // ★人魚を捕獲済みか否かを入れる変数
20.   loadMermaids(); // ★人魚の情報を読み込む
21.   var getImg = document.getElementById("getImg"); // ★img要素の取得
22.   var canvas = document.getElementById("cap"); // ★捕獲済人魚を入れるcanvas要素の取得
23.   var context = canvas.getContext("2d"); // ★contextの取得
24.   context.fillStyle = "rgb(153, 217, 234)"; // ★塗りつぶす色をターコイズにする

```

このアプリは、人魚を配置する場所の情報を、index.html とは別のファイルで提供することになっています。そこで、Visual Studio Code（または好みのエディタ）を立ち上げて、以下のようなファイルを作ります。ファイ

ルは、C:\xampp\htdocs フォルダの中の「gps3」フォルダの中に、「mermaids.json」として保存してください。

```

1. [
2.   {
3.     "name": "センタースクエア",
4.     "lat": 35.157723,
5.     "lng": 137.031170,
6.     "img": "0.png"
7.   },
8.   {
9.     "name": "研究棟前",
10.    "lat": 35.157263,
11.    "lng": 137.032172,
12.    "img": "1.png"
13.  },
14.  {
15.    "name": "はっぴー広場",
16.    "lat": 35.158301,
17.    "lng": 137.032059,
18.    "img": "2.png"
19.  },
20.  {
21.    "name": "ASUテラス",
22.    "lat": 35.156180,
23.    "lng": 137.031595,
24.    "img": "3.png"
25.  },
--

```

ここでは、私の勤め先である愛知淑徳大学のキャンパス内の場所 5箇所の緯度と経度を指定しています。愛知淑徳大学に関係の無いかたは（無いかたのほうが多いですね...）、自宅や職場の近くなどの場所の緯度と経度を指定しましょう。任意の場所の緯度と経度は Google マップで調べることができます。

まずはパソコンで [Google マップにアクセス](#)します。人魚を配置したい場所周辺に地図を移動させて、人魚を配置したい場所をクリックしてください。以下の図では場所を特定しやすいようにマップを航空写真にしていますが、地図のままでも構いません。



Google マップで人魚を配置したい場所をクリック

すると、クリックした場所の住所と緯度と経度が表示されます。この緯度と経度をコピーして、mermaids.json の緯度(lat) と経度(lng) の部分に貼り付けてください。また、その場所がどんな場所かを name: の横の " " の中に書いておきましょう。GPS を使うので、できるだけ空が見える広い場所を指定します。

この時、**私有地など他の人に迷惑がかかる場所や、道路・線路など危険な場所の近くは絶対に指定してはいけません！**誰でも行くことができ、道路・線路などから離れていて、近くに階段などもない、公園などが最適です。できれば、公園であっても、利用者の少ない場所のほうが他の利用者に迷惑がかりません。私有地や危険

な場所にポケモンが出たり、公園の他の利用者が迷惑だといって「ポケモンGO」で大きな社会問題になっていましたよね？

なお「JSON」というファイル形式については後ほど解説します。

動作確認

このアプリではさらに 5 個の人魚の画像ファイルが必要です。以下の 5 個の画像をダウンロードしてください（パソコンのブラウザで画像を右クリックして「名前をつけて画像を保存」などの方法で）。ダウンロードした 5 個の画像を「gps3」フォルダの中に入れてください（index.html と同じ階層に）。



あとは、[ステップ1と同じやり方](#)で動作確認してください。

Android の Firefox が使えるなら、アドレス欄に以下のように入力します。以下の「10.11.52.81」の部分はパソコンで ipconfig して調べた IP アドレスです。

- 10.11.52.81/gps3

自分のホームページスペースを持っていて、かつそこが https 接続に対応していれば...

- **https://**自分のHPスペースのアドレス/gps3

自分のホームページスペースを持っていない人、および、持っていない人も https 接続に対応していなければ、パソコンで [Netlify Drop](#) などの無料ホスティングサービスに「gps3」フォルダをアップして、表示されたアドレスをスマホのブラウザで開きます。この時、**アドレスの先頭に「https://」と入力してください。**

スマホで以下のように表示されます。



アクセスすると

さあ、人魚のいる場所に行ってみましょう。人魚のいる場所に近づくと、そこにいる人魚が大きく表示されて、画面の下の枠の中に人魚をゲットすることができます。人魚をタップすると地図の画面に戻ります。

この時、**危険ですから、自動車や自転車などで動き回ったり、動きながらスマホを見るのは絶対にやめましよう。**



人魚をゲット

ゲームをもう一度やる場合は、ブラウザでアプリをリロード（再読込）してください。

解説

18～25行目は様々な変数の宣言です。それぞれ何なのかは自分でプログラムの中を追ってみましょう。

gps2 から主に加わったのは、66行目からの loadMermaids() 関数、79行目からの placeMermaids() 関数、98行目からの calcDistance() 関数です。

loadMermaids() は、人魚の居場所の情報を書いた「mermaids.json」ファイルを読み込む関数です。プログラムの最初のほう、20行目でこの関数を呼び出しています。

```

65. // ★人魚の情報を読み込む loadMermaids 関数
66. function loadMermaids() {
67.     var req = new XMLHttpRequest(); // サーバのファイルを読む XMLHttpRequest
68.     req.addEventListener("readystatechange", () => { // 準備状態に変化があった時の処理
69.         if(req.readyState === 4 && req.status === 200) { // データ受信が正常に完了したら
70.             mermaids = JSON.parse(req.responseText); // 読み込んだJSONデータを整形して人魚データに入れる
71.         }
72.     });
73.     req.open("GET", "mermaids.json"); // リクエストを設定
74.     req.send(); // リクエストを送る
75. }

```

Web アプリでサーバに置かれたテキストファイルを読み込むには、XMLHttpRequest というオブジェクトを使います。ここでは req という名前でこのオブジェクトを作って（67行目）、req.open() の () 内でファイル名を指定して（73行目）、req.send() で読み込みを実行しています（74行目）。req.send() を送るとサーバがファイルを読み込んで返事を返してきて、「readystatechange」というイベントが発生します。「readystatechange」イベントが発生したら（68行目）、読み込んだファイルの内容 req.responseText を mermaids という変数に入れています（70行目）。

ここで「JSON 形式」について少し解説しておきます。mermaids.json ファイルは、以下のような構造をしています。

```

1. [
2.   {
3.     "name": "センタースクエア",
4.     "lat": 35.157723,
5.     "lng": 137.031170,
6.     "img": "0.png"
7.   },
8.   { ...

```

ここで、{ } で囲まれた範囲が 1個の「データ」です。さらに、「name」や「lat」という文字列をデータの「key（キー）」と呼び、「センタースクエア」や 35.177723 を「value（値）」と呼びます。「0番目のデータの Key が lat の valueは 35.177723」となります。プログラム中で、mermaids[0].lat とすることで、35.177723 という値が取り出せます。

今回の mermaids.json の中では、{ } で囲んだデータが 5個あります。{ } で囲んだデータとデータの間はカンマ (,) で区切ります。また、最初と最後に [] がありますが、これは「データの配列」であることを示しています。

少しややこしいかもしれませんが、こういう形式で様々な値がセットになったデータを記述する形式を JSON (JavaScript Object Notation) 形式といいます。JavaScript のプログラムではよく使われますし、近ごろは JavaScript 以外の言語でもこの形式でデータを扱うことが増えています。

次に placeMermaids() は、人魚を地図上に配置する関数です。地図を初期化する initMap() 関数の中（58行目）から呼び出しています。

```

77. // ★人魚を地図上に配置する placeMermaids 関数
78. function placeMermaids() {
79.     var mermaidMark = []; // 人魚マーカーの配列
80.     for(var i = 0; i < mermaids.length; i++) { // 全ての人魚について
81.         var pos = new google.maps.LatLng(mermaids[i].lat, mermaids[i].lng); // 人魚の位置を設定
82.         var img = "0.png"; // 画像の設定

```

```

82.     var img = {
83.         url: mermaids[i].img,           // 画像の設定
84.         scaledSize: new google.maps.Size(60, 60) // 画像ファイル名
85.     };                                     // 画像を縮小表示
86.     mermaidMark[i] = new google.maps.Marker({ // 人魚のマーカーを作成
87.         map: gmap,                       // gmap の上に表示する
88.         position: pos,                   // pos の位置に
89.         icon: img,                       // アイコンを設定
90.         title: mermaids[i].name         // タイトルを設定
91.     });
92.     captured[i] = false;                 // 捕獲済み状態を全てfalseにする
93. }
94. }

```

人魚の画像を Google マップの「マーカー」として置いています。自分がいる場所にはよく見る「ピン」の形をしたマーカーを置きました。これが Marker オブジェクト標準の画像なのですが、Marker オブジェクトのオプションで「icon」属性に画像ファイルを指定することで（89行目）、画像をマーカーにすることができます。

最後に、このゲームで最も重要な、自分と人魚の距離を計算する calcDistance() 関数です。

```

96. // ★自分と人魚との距離を計算する calcDistance 関数
97. function calcDistance(lat, lng) {
98.     var distance = []; // 距離を入れる配列
99.     var myPos = new google.maps.LatLng(lat, lng); // Googleマップの LatLng オブジェクトを作成
100.    for(var i = 0; i < mermaids.length; i++) { // 全ての人魚について
101.        var pos = new google.maps.LatLng(mermaids[i].lat, mermaids[i].lng); // 人魚の位置を設定
102.        distance[i] = google.maps.geometry.spherical.computeDistanceBetween(myPos, pos); // 距離を求める
103.        // 捕獲の判定と捕獲した時のエフェクト
104.        if(distance[i] < 20 && captured[i] === false) { // 距離が20m未満、かつ、まだ捕獲していないなら
105.            captured[i] = true; // 捕獲済にする
106.            getImg.src = mermaids[i].img; // 捕獲された人魚の画像を設定
107.            context.drawImage(getImg, i * 60, 0, 60, 60); // 捕獲済の枠に人魚の画像を表示
108.            getImg.hidden = false; // img要素を表示
109.            mapDiv.hidden = true; // 地図を非表示
110.            getImg.addEventListener("click", () => { // img要素がクリックされたら
111.                getImg.hidden = true; // img要素を非表示
112.                mapDiv.hidden = false; // 地図を表示
113.            });
114.        }
115.    }
116. }

```

まず、自分の位置 myPos と、各人魚の位置 pos を求めています。自分と各人魚の距離（単位: m）を計算しているのは 102行目の google.maps.geometry.spherical.computeDistanceBetween(myPos, pos) です。とても長い命令ですが、Google マップに備わっている、二点間の距離を計算するメソッドです。

このメソッドを使うには、Google マップを呼び出す <script> タグの中で、「libraries=geometry」という記述を書き加える必要があります。

```

122. <!-- ★距離を求めるライブラリを使うために末尾に「&libraries=geometry」を追加 -->
123. <script src="https://maps.googleapis.com/maps/api/js?key=API_KEY&callback=initMap&libraries=geometry">
124. </script>

```

距離が計算できたら、距離が 20 m 未満で、かつ、まだ捕まえていなかったら（104行目）、そこにいた人魚の画像を img 要素に入れて、はじめ隠しておいた（hidden 属性をつけておいた）img 要素を表示（hidden = false）しています。また、地図の入った div 要素を非表示（hidden = true）にしています。

参考

完成品を[こちら（大学）](#)または[こちら（Netlify Drop）](#)に置いてありますからスマホで開いてみてください。

補足

ここで使った人魚の画像は[GATAG | フリーイラスト素材集](#)からダウンロードした画像をもとに、サイズ調整や背景の加工などを行いました。

もし自分で二点間の距離を計算で求めようとする、例えば[ここに書いてあるような](#)とても複雑な計算が必要です。Google マップはこのようなことを簡単に（しかも無償で）できるとも便利なものなのです。

[▲TOP](#)

まとめ

スマホの GPS を使ったアプリ、いかがでしたか？

緯度と経度については「中学校で習ったけどそんなものいつ使うの？」だったかもしれませんが、スマホではこんなふうになら日常的に使われています。また、Google マップというのは単なる「地図アプリ」ではなくて、位置情報を使った様々な便利なことに使えるサービス、ということが垣間見えたと思います。ユーザの位置が分かると、道案内やカーナビだけでなく、その場所の天気、その場所の近くにあるお店、あなたの近くにいる別のユーザ、なども分かります。応用が広がりますね。

大ヒットした「ポケモンGO」は今回の GPS に加えて、ジャイロセンサや AR の技術を使っています。なかでもいちばん重要なのが GPS でユーザの位置を測る機能です。mermaids.json で人魚の位置を設定しましたが、ポケモンGO でもこれに似たような方法でポケモンが出る場所を決めていると考えられます（詳しいことは分かりませんが）。また、例えばあるメーカーのあるカーナビでは、一時停止標識がある場所をこのように指定しておいて、近づいたら、見のがさないように注意を促すような機能もあります。位置情報は使い方次第で暮らしをとても便利にしてくれます。

ただ、途中でも書いたように、ユーザが居る場所が分かるということは、プライバシーを侵害するような悪用もされかねないものです。そのため各スマホの OS やブラウザのメーカーは https 接続のサイトでの利用に制限するなど、いろいろ難しい技術であることも事実です。ポケモンGO では、私有地や危険な場所にポケモンが置かれたとか、あまりの面白さに自動車の運転中にプレイしてしまうなど、大きな社会問題にもなりました。位置情報を使うサービスを作る人は、そのコンテンツの影響やセキュリティを慎重に検討する必要があります。とはいえ、怖がっているだけでなく、技術を使いながら、上手に使っていく方法を探しましょう。

まずは個人としてゲーム作りなどで楽しんで、慣れて、どんどん応用を考えてみましょう！

補足

上で作ったコードはすべて[こちら \(GitHub\)](#) に置いてありますから、参考にしてください。