２０１４．６．３０

医療者のための情報技術入門

第１０回　プログラムがはたらくしくみを学ぶ（３）

日紫喜　光良

概要

１．はじめにー具体例からー

２．Javascript のプログラミング入門

----------ここから--------

３．足りないものは借りてくるーJavascript のライブラリ（→現在地の住所を調べる）

４．仕事は人にやらせるーサーバとブラウザの役割分担（→現在地をメールで知らせる）

３．Google Maps API を使って現在地の住所を調べる

3.1 はじめに

一般的に、あるコンピュータプログラム(ソフトウェア)の機能や管理するデータなどを、外部の他のプログラムから呼び出して利用するための手順やデータ形式などを定めた規約をアプリケーションプログラミングインターフェース（Application Programming Interface）略して API という。プログラミングを行う際に API を使うと作業が抽象化・単純化できる。

ブラウザは、ブラウザ上で動くプログラムのための API を用意している。新しいブラウザの多くは、HTML5 の規格で定められた API に対応し、ブラウザ上でブラウザの操作を行う Javascript のプログラムを動作できる。先週紹介したプログラムの `play()`, `pause()` 等のメソッドは、HTML5 に準拠したブラウザが、オーディオプレーヤーオブジェクトを操作するために提供している API である。

Google Maps（http://maps.google.com/）は、グーグルが提供している地図情報サービスで、街路地図、衛星写真、ストリートビュー、ルート検索機能などの機能がある。これらの機能をグーグルのサイトで利用できるだけでなく、Google Maps API（現在は ver. 3）を用いて、自作のプログラムの中で呼び出して利用することができる。

今回は、まず、無線 LAN・公衆 WiFi・携帯電話基地局・GPS・IP アドレスなどから地理情報をブラウザで取得するための統一された方法である HTML5 の Geolocation API を利用して、現在地の緯度・経度を取得する。次に、Google Maps API の Reverse geocoding API を利用して、緯度・経度から大まかな住所を取得する。

3.2 現在地の緯度と経度を取得する。

次のプログラムに拡張子が html であるような適当な名前を付けて FireFox ブラウザで開くと、緯度と経度を表示する。

（注：Google Maps API を用いたプログラムをローカルな環境（手元のコンピュータ上で実行する）でテストするには、FireFox ブラウザが最適であるように思われる。サーバ上にアップすれば、Chrome でよい。）

```
<!DOCTYPE HTML>
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="robots" content="noindex, nofollow" />
<title>現在地を住所にする</title>
<script>
var latitude,longitude,address,accuracy;
window.onload = function(){
   navigator.geolocation.getCurrentPosition(is_success,is_error);
}

function is_success(position) {
    latitude = position.coords.latitude;
    longitude = position.coords.longitude;
    accuracy = position.coords.accuracy;
    result = "Latitude = " + latitude + "; Longitude = " + longitude + "; Accuracy = " + accuracy + " meters.";
    document.getElementById('message').innerHTML = result;
}
function is_error(error) {
    var result = "";
    switch(error.code) {
        case 1:
            result = '位置情報の取得が許可されていません';
        break;
        case 2:
            result = '位置情報の取得に失敗';
        break;
```

```
29          case 3:
30              result = 'タイムアウト';
31          break;
32      }
33      document.getElementById('message').innerHTML = result;
34  }
35
36  </script>
37  </head>
38  <body>
39  <div id="message"></div><br>
40  </body>
41  </html>
```

このプログラムのすべての動作は 9~11 行目に書いてある。

```
 9  window.onload = function(){
10    navigator.geolocation.getCurrentPosition(is_success,is_error);
11  }
```

window.onload = function () { … }は、「ページを読み込んだら、…を行う」という意味である（前回参照のこと）。

そして、navigator.geolocation.getCurrentPosition とは、「navigator オブジェクトのプロパティ（属性）である geolocation オブジェクトの getCurrentPosition メソッド」を特定するための書き方である。ここで、navigator オブジェクトは、サイトを訪れた（つまりこのファイルを開いた）ユーザーのブラウザ情報や PC 環境情報（OS は何を使っているかなどの情報）を取得する。また、navigator オブジェクトのプロパティでもある geolocation オブジェクトは、現在位置の取得をおこなう。getCurrentPosition メソッドには、それぞれ現在位置の取得に成功した時、失敗した時に実行する、２つの関数への参照（関数の名前）を与える。

成功した時に実行する関数 is_success()には、自動的に引数として位置情報を含むオブジェクト Position（プログラムの中で使うとき、名前は何でもよいが、仮に position という名前にしておく）が与えられる（13 行目）。そして、そのプロパティである coords オブジェクトのプロパティから、緯度（latitude プロパティ）、経度（longitude プロパティ）、精度を円の半径（単位はm）であらわしたもの（accuracy プロパティ）を得ることができる（14～16 行目）。

17 行目は、メッセージを作るためにこれらの変数をつなげるための命令である。18 行目は、39 行目で定義した message という名前の 要素に result を書きだすための命令である。

```
13 function is_success(position) {
14     latitude = position.coords.latitude;
15     longitude = position.coords.longitude;
16     accuracy = position.coords.accuracy;
17     result = "Latitude = " + latitude + "; Longitude = " + longitude + "; Accuracy = " + accuracy + " meters.";
18     document.getElementById('message').innerHTML = result;
19 }
```

実行結果は、次のようになる。

Latitude = <緯度の値>; Longitude = <経度の値>; Accuracy = <誤差の値> meters

3.3 現在地の住所を取得する。

前の節では結果の書き込みを is_success 関数で行っていたが、それを、gmap_message 関数で行うことにする。

```
function is_success(position) {
    latitude = position.coords.latitude;
    longitude = position.coords.longitude;
    accuracy = position.coords.accuracy;
    //result = "Latitude = " + latitude + "; Longitude = " + longitude + "; Accuracy = " + accuracy + " meters.";
    //document.getElementById('message').innerHTML = result;
    gmap_message(latitude,longitude,accuracy);
}
```

新たに作成する関数 gmap_message で作成するメッセージには、緯度と経度から得た住所も加えることにする。この関数は次のようになる。

```
function gmap_message(lat,lng,acc) {
    var geocoder = new google.maps.Geocoder();
    var latLng = new google.maps.LatLng(lat,lng);
    var msg = "";
    geocoder.geocode({'latLng':latLng},function(results,status){
        if (status == google.maps.GeocoderStatus.OK) {
```

```
            msg     = '現在地の取得に成功<br>';
            msg += '経度：' + lat + '<br>';
            msg += '緯度：' + lng + '<br>';
             msg += '精度（メートル）：' + acc + '<br>';
            msg += '住所：' + results[0].formatted_address + '<br>';
            document.getElementById('message').innerHTML = msg;

        } else {
            console.log(status);
        }
    });
}
```

ここで中心になるのは、geocoder オブジェクトである。このオブジェクトを生成したり操作したりするためのプログラム群（ライブラリ）をインターネット上から取得するために、Web ページのヘッダー内（<header>と</header>の間）に次の宣言を 1 行加える必要がある。この宣言は、プログラム本体を含む<script>タグよりも前に行う必要がある。

```
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false&language=ja"></script>
```

さて、geocoder オブジェクトは、google.maps.Geocoder()メソッドによって生成される。オブジェクトを生成するメソッドをコンストラクタ constructor という。コンストラクタを用いるときは、new という命令といっしょに使う。

```
var geocoder = new google.maps.Geocoder();
```

緯度と経度から住所を取得するには、geocoder.geocode メソッドを用いる。1 番目の引数に、緯度と経度とからなるオブジェクトを与える。このオブジェクトは、コンストラクタ google.maps.LatLng()に引数として緯度と経度を与えることで生成される。

geocode メソッドは、逆に、住所から緯度と経度を求めるのにも使える。例では 1 番目の引数が{'latLng':latLng}となっているが、これを{'address':<住所文字列>}とすると、緯度と経度からなるオブジェクトを得ることができる。

2 番目の引数には、geocode メソッドが実行された後に実行される関数が入る。この関数には自動的に results と status の 2 つの引数が与えられる。results には geoode メソッドの実行結果が、status には終了状態（正常かどうか）が、それぞれ入る。

results オブジェクトは、オブジェクトの配列である。配列の何番目の要素を選ぶかによって、住所の詳細度が異なる。それぞれの要素の formatted_address プロパティに、住所

の文字列が格納されている。

さて、この関数を実行する際には緯度、経度、精度をあらわす３つの引数を与える必要がある。これらは関数 is_success()内で発生するので、もしもこれらを表す変数がこの関数の内部で宣言されていたら、同じ名前の変数を使っても、関数の外でその値を使うことはできない。しかし、この関数の外で既に宣言されていれば、この関数より後で別の関数の中で使われる同じ名前の変数は、同じ値を指す。そこで、is_success()関数を使用するよりも先に大域（グローバル）変数として latitute, longitude, accuracy を宣言しておく必要がある（8行目）。

3.4 現在地を示すマーカーを地図上に立てる

Google Map の表示をおこなうために、次のような拡張をおこなう。

まず、HTML document の構造に、地図を表示する領域 を設定する。

```
<body>
<div id="message"></div><br>
<div id="map"></div>
</body>
```

同時に、HTML 文書のヘッダー内に、次のような CSS 形式で、「スタイルシート」を記述する。

```
<style>
        #map {width: 500px; height: 380px; border: 1px solid #666; float: left;}
</style>
```

次に、is_success 関数を、その中に次の A),B)の手続き（関数）を追加することで、拡張する。

A) gmap_draw 関数。緯度と経度を与えられると、上で追加した要素名 map の 要素に、その地点を中心とする地図オブジェクトを表示する。次のように、地図オブジェクトはコンストラクタ google.maps.Map()によって生成される。

```
function gmap_draw(lat,lng){
   var op = {
                zoom:13,
                center:new google.maps.LatLng(lat,lng),
                mapTypeId:google.maps.MapTypeId.ROADMAP
   };
   map=new google.maps.Map(document.getElementById("map"),op);
```

```
}
```

ここで、op オブジェクトの３つのプロパティ（zoom, center, mapTypeId）は次のような機能をもつ。

zoom: 地図のズームレベル（縮尺）をさす。数字が大きいほど、縮尺が大きく（表示されるものが大きく）なる。

center: 中心の緯度と経度をさす。プロパティの値は、緯度と経度を合わせたオブジェクトが入る必要がある。そこで、コンストラクタ google.maps.LatLng()によって生成している。

mapTypeID: 市街地図、航空地図等の地図の種類を選択する。

B) placeMarker 関数。地図と緯度と経度を与えられると、マーカーオブジェクトを生成し、地図上のその地点にマーカーを置く。次のように、マーカーオブジェクトはコンストラクタ google.maps.Marker()で生成される。

```
function placeMarker(map,lat,lng){
  var latLng = new google.maps.LatLng(lat, lng);
  var label = "Here am I.";
  marker = new google.maps.Marker({position:latLng, map:map, title:label});
}
```

コンストラクタ google.maps.Marker()は３つのプロパティ(position, map, title)を必要とする。

position: マーカーを置く位置をあらわす。緯度と経度がセットになったオブジェクトであることが必要であるので、google.maps.LatLng()コンストラクタで生成する。

map: gmap_draw 関数で生成した地図オブジェクトと同一である必要がある。ところが、関数の中で宣言された変数の名前が有効なのは、宣言された関数の中だけである。このため、あらかじめ、gmap_draw 関数の外で、var map; として、map の宣言をしておく。gmap_draw 関数の map の頭に var が宣言されていなかったのは、そのためである。そこでは、すでに宣言された map に代入し（参照を渡し）たにすぎない。

title: マーカーの上にカーソルを置くと表示される、ラベルとしての役割をもつ。

前節の gmap_message 関数の追加とあわせ、これら３つ（gmap_message, gmap_draw, placeMarker）の関数を追加した結果、全体は次のようなプログラムになる。

```
1 <!DOCTYPE HTML>
2 <html lang="ja">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5 <meta name="robots" content="noindex, nofollow" />
```

```
<title>現在地を住所にする</title>
<style>
        #map {width: 500px; height: 380px; border: 1px solid #666; float: left;}
</style>
<script                                                    type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false&language=ja"></script
>
<script>
var latitude,longitude,address,accuracy;
var map;
var marker;
window.onload = function(){
   navigator.geolocation.getCurrentPosition(is_success,is_error);
}

function is_success(position) {
    latitude = position.coords.latitude;
    longitude = position.coords.longitude;
    accuracy = position.coords.accuracy;
    //result = "Latitude = " + latitude + "; Longitude = " + longitude + "; Accuracy
= " + accuracy + " meters.";
    //document.getElementById('message').innerHTML = result;
    gmap_message(latitude,longitude,accuracy);
    gmap_draw(latitude,longitude);
    placeMarker(map,latitude,longitude);
}
function is_error(error) {
    var result = "";
    switch(error.code) {
       case 1:
          result = '位置情報の取得が許可されていません';
       break;
       case 2:
          result = '位置情報の取得に失敗';
       break;
       case 3:
```

```
39            result = 'タイムアウト';
40        break;
41    }
42    document.getElementById('message').innerHTML = result;
43 }
44
45 function gmap_message(lat,lng,acc) {
46    var geocoder = new google.maps.Geocoder();
47    var latLng = new google.maps.LatLng(lat,lng);
48    var msg = "";
49    geocoder.geocode({'latLng':latLng},function(results,status){
50        if (status == google.maps.GeocoderStatus.OK) {
51            msg    = '現在地の取得に成功<br>';
52            msg += '経度：' + lat + '<br>';
53            msg += '緯度：' + lng + '<br>';
54             msg += '精度（メートル）：' + acc + '<br>';
55            msg += '住所：' + results[0].formatted_address + '<br>';
56            document.getElementById('message').innerHTML = msg;
57
58        } else {
59            console.log(status);
60        }
61    });
62 }
63
64 function gmap_draw(lat,lng){
65    var op = {
66                zoom:13,
67                center:new google.maps.LatLng(lat,lng),
68                mapTypeId:google.maps.MapTypeId.ROADMAP
69    };
70    map=new google.maps.Map(document.getElementById("map"),op);
71 }
72
73 function placeMarker(map,lat,lng){
74   var latLng = new google.maps.LatLng(lat, lng);
```

```
75      var label = "Here am I.";
76      marker = new google.maps.Marker({position:latLng, map:map, title:label});
77  }
78  </script>
79  </head>
80  <body>
81  <div id="message"></div><br>
82  <div id="map"></div>
83  </body>
84  </html>
```

４．　現在地を人にメールで知らせる

現在地を表示した後、自分と相手の E-mail アドレスを入力してボタンをクリックすれば、相手に現在地がメールされるように改造するにはどうすればよいか？ブラウザ自身から直接メールを発信することはできない。ブラウザからサーバに対し住所を含むメッセージを送信し、サーバを介して相手のメールアドレスに対して発信することになる。

サーバ上で、そのためのプログラムを作成する必要がある。そのような目的のプログラミング言語として、PHP, Perl, Ruby, Node.js などがあり、最初の３つが特によく用いられる。外部プログラムと連携できるように設定された Web サーバプログラムと、これらのいずれかの言語を処理実行できるようなサーバが必要である。

ホームページを置くことができ、かつ、PHP を Web ページの中で利用できるようなレンタルサーバの契約をする場合の費用は、さまざまである。例えば料金は「さくらインターネット」の場合、スタンダードプラン年間一括払いとすると 5000 円強であり、他に初期費用が 1000 円程度かかる。また、お試し利用が 2 週間可能である。

PHP を用いた場合の、ブラウザ上で開く Web ページの変更点の要点は次のようになる。

まず、HTML 文書の要素は次のようになる。from と to が、E-mail アドレスを入力するテキストボックスである。また、send が送信ボタン、response がサーバから返されたメッセージの表示スペースである。

```
<body>
<div id="message"></div><br>
Your E-mail address: <input type="text" id="from"></input><br>
To: <input type="text" id="to"></input><br>
```

```
<button id="send">Send my locaion</button><br>
<div id="response"></div><br>
<div id="map"></div>
</body>
```

is_success()関数の中に、「実行ボタン send をクリックしたときはサーバにメッセージを送信するための関数 sendAddress を呼び出す」ことの宣言を追加する。この書き方では、sendAddress に引数としてあらかじめ latitude や longitude を与えておくことはできない。

```
function is_success(position) {
    latitude = position.coords.latitude;
    longitude = position.coords.longitude;
    var accuracy = position.coords.accuracy;
    gmap_message(latitude,longitude,accuracy);
    gmap_draw(latitude,longitude);
    placeMarker(map,latitude,longitude);
    $("#send").on('click',sendAddress);
}
```

このように jQuery ライブラリを使うので、プログラムの本体を含む<script>タグの外側に、次の宣言を 1 行加える。

```
<script type='text/javascript'
src='http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js'></script>
```

sendAddress 関数の本体を定義する。入力は、グローバル変数として定義された latitude, longitude, accuracy, address の他に、入力欄 from ならびに to から $(“#要素の ID”).val()によって取得したメールアドレスで、サーバへの送信は、

```
$.post('送信相手のプログラム', {送信される項目名:その値, 送信される別の項目名:その値,…},
function(data,status){処理});
```

という形をとる。ここで、data ならびに status はそれぞれ、サーバから返される HTML データと処理が成功したかどうかの情報、をあらわす。従って、次のようになる。

```
function sendAddress(){
  //console.log(latitude+":"+longitude);
  var from = $("#from").val();
  var to = $("#to").val();
```

```
  var msg = address + " (with accuracy "+ accuracy + " meters)";
  $.post("mail-latlng.php",
     {from:from,to:to,msg:msg,lat:latitude,lng:longitude},
     function(data,status){
        $("#response").text(data + ":" + status);
     }
  );
}
```

次に、サーバー側では、リクエストのメソッドが POST であるかどうかの判定($_SERVER["REQUEST_METHOD"] == "POST") が真であれば、$from = $_POST["from"]などの形で、ブラウザから送られたメッセージから項目を取り出す。次に mail()関数によって相手にメールを送る。ただし、セキュリティ上の理由（インジェクションを防ぐ）で、メールアドレスとメッセージ本体は htmlspecialchars()関数で<を<に、>を>に、などの文字変換が必要である。さらに、from, to については isset()関数で空欄でないことを確認し、次の spamcheck()関数でメールアドレスの形式であることをチェックする。

```
function spamcheck($field) {
  // Sanitize e-mail address
  $field=filter_var($field, FILTER_SANITIZE_EMAIL);
  // Validate e-mail address
  if(filter_var($field, FILTER_VALIDATE_EMAIL)) {
    return TRUE;
  } else {
    return FALSE;
  }
}
```

また、メッセージ本体の長さは 70 文字までと決まっているので、wordwrap()関数で途中で切る。通常の携帯電話（いわゆるガラケー）で認識できるように、文字コードを Shift-JIS に変換する。

```
$message = mb_convert_encoding($message, "SJIS-win", "UTF-8");
```

参考

サーバ側の PHP プログラムについては、「W3schools.com PHP Tutorial」（http://www.w3schools.com/php/）の”PHP E-mail”, “PHP Secure E-mail”を参考のこと。w3schools.com の他の言語、例えば HTML, Javascript, jQuery, のチュートリアルおよびタグリファレンスも参考になる。