ななちゃんのIT教室

マッシュアップに挑戦の巻

by nara.yasuhiro@gmail.com

ななちゃんが
マッシュアップに挑戦するという お話

第 1.0 版　2017 年 5 月 7 日

フリー素材
http://freeillustration.net

注意事項

マッシュアップでは外部サイトのサービスを利用します。本資料では、無料で利用できるサイトを利用します。無料で利用できるサイトの仕様は随時変化する可能性があります。本資料の内容の「考え方」は恒久的でも、外部サイトの仕様変更にともない、例題プログラムが正しく動作しなくなる可能性があります。また、業務に使うプログラムを開発する場合は、外部サイトの「利用規約」を確認の上、必要なら、有料サービスの利用を検討してください。

もくじ

第1回　マッシュアップとは

なな：今回のお題の「マッシュアップ」って何？

先生：複数の外部サービス（第三者サービス）を寄せ集めて、ひとつのプログラムにまとめあげることよ。複数のジャガイモをつぶして、ポテトサラダにする「マッシュポテト」の感覚ね。

なな：どんな使い方をするの？

先生：米国で、アンケート結果を集計するウェブサイトを作った成功例で有名になったのよ。開発に数年かかると言われていたウェブサイトが、たった数ヶ月で実現してしまったの。地震などの自然災害が起きた地区で、消息情報を共有したり、復興に必要な情報を提供するウェブサイトをきわめて短時間で立ち上げるといった分野でも大きな成果をあげているわ。今回は、 その原理を説明した上で、 郵便番号検索（郵便番号から住所を調べる）の例を体験してみるわね。

なな：なんか難しそうね。

先生：原理は一応説明するけど、理解できなくても構いません。外部サービスを利用する「サンプルプログラム」を入手して、「プログラムのここに郵便番号を書けば、ここで住所が得られる」ということを見抜き、「じゃあ、input タグから郵便番号を入力したら、それをここに挿入すれば良い」というようなことを考え、プログラムとして表現できること（カスタマイズできること）が重要ね。基本的な JavaScript の知識だけで十分よ。あとは練習あるのみね。

なな：郵便番号検索（郵便番号から住所を調べる）の例を体験するのね。

先生：そうね。そして、住所情報に含まれる緯度、経度情報から、地図を表示してみましょう。ポイントは、郵便番号を検索するサービスと、地図を表示するサービスの提供者が異なるということ。別々の業者が提供するサービスを組み合わせて、自分が必要とするひとつのサイトに組み立てるのがマッシュアップのだいご味なのよ。

なな：別々の業者のサービスって、うまくつながるの？

先生：大丈夫よ。素材となるサービスのプログラム詳細は知る必要がなく、サービスを呼び出す手順と、そのサービスから送り返されるデータの形式、Web API（Application Program Interface）> と呼ぶんだけど、それだけに注目すれば良いの。

なな：外部サービスって、使うのにお金を払わなくて良いの？

先生：良い質問ね。外部サービスの多くは有償で提供されていて、契約すると、契約者の特定のURLからのアクセスが許可され、アクセス度数に応じて課金されるの。でも、無償のサービスも提供されているから安心してね。ユーザに使い慣れてもらい、もっと高機能な有償版に乗り換えてもらうきっかけにするとか、ユーザがどんな使い方をするか、情報を収集するのか調べるとか、バグなどの問題点を洗い出して、有償版を開発するための情報を得るとかなどの理由で無償サービスが成り立っているのね。勉強には、このような無償サービスを利用するのが良いと思います。

なな：無償サービスで気をつけることはないの？

先生：無償サービスでは、「バグなどによって不利益をこうむっても弁償はしてもらえない」、「予告なしに提供が中止されたり、仕様が変更されて、今まで動いていたユーザプログラムが動かなくなっても補償してもらえない」などの制約があるのが要注意ね。制約は「使用許諾条件」に書かれていて、それを承認した上で使用することになるの。無償サービスは、勉強には最適だけど、業務利用には不向きね。使用許諾条件に、商用利用の禁止が含まれている場合もあるわ。とにかく、サービス利用前に使用許諾条件を読むことが必要ね。

なな：了解！

第2回　マッシュアップの原理（問題点）

先生：まず郵便番号検索外部サービス「Heartrails」の機能を確認しましょう。ななちゃん、ブラウザの URL 欄に、
http://geoapi.heartrails.com/api/json?method=searchByPostal&postal=2520816&jsonp=callbackf
と入力してみてね。「2520816」の部分が、調べたい郵便番号になっているの。

なな：画面に、こんなのが表示されたわ。

```
callbackf({"response":{"location":[{"city":"¥u85e4¥u6ca2¥u5e02","city_kana":
"¥u3075¥u3058¥u3055¥u308f¥u3057","town":"¥u9060¥u85e4","town_kana":
"¥u3048¥u3093¥u3069¥u3046","x":"139.435752","y":"35.384784","prefecture":
"¥u795e¥u5948¥u5ddd¥u770c","postal":"2520816"}]}})
```

先生：そうね。日本語の文字は、通信を混乱させる場合があるので、16進数の文字コードになっているの。これを、JavaScript を使って解釈すると、こんなふうになるの。

```
callbackf({"response":{"location":[{"city":"藤沢市","city_kana":"ふじさわし",
"town":"遠藤","town_kana":"えんどう",
"x":"139.435752","y":"35.384784","prefecture":"神奈川県","postal":"2520816"}]}})
```

なな：たしかに、、「藤沢市」とか、「神奈川県」とか、郵便番号に対する地名情報が含まれているみたいね。

先生：JavaScript プログラムから、外部サイトにアクセスをして、このような情報を受け取るには、以前は、下記のような記述をすれば良かったの。

```
var request = new XMLHttpRequest();
request.onreadystatechange = checkStatus;
request.open("GET", "http://geoapi.heartrails.com/api/json?method=searchByPostal&postal=2520816&jsonp=callbackf", true);
request.send(null);

function checkStatus(){
   if (request.readyState == 4 && request.status == 200) { alert(request.responseText); }
}
```

なな：「良かった」？　過去形ね。

先生：そう。ウェブサーバに不法侵入して、ウェブページの JavaScript プログラムを書き換え、悪意の含まれる犯人サイトにジャンプさせるような事件が起こったの。ユーザは、ちゃんとしたウェブページにアクセスしているつもりが、実は犯人のウェブページにアクセスしていて、入力したクレジットカードの番号を盗まれたり、ウィルスに感染させられたり。こういう、サイト外にジャンプしてしまうことを「リダイレクション」といいます。

なな：怖いわね。

先生：こういう事件が発生してから、JavaScript ではリダイレクションを禁止するようになったの。

なな：JavaScript が禁止したの？

先生：正確には、ブラウザが、JavaScript の、リダイレクションに関する命令を実行しないで、実行時エラーにするようになったということ。こういうのを、「同一生成元ポリシー」というの。ダウンロードしてきた JavaScript プログラムが、ダウンロード先以外のサイトと通信しようとすると、それを抑制するという制限ね。

なな：じゃあ、マッシュアップは実現できないっていうこと？

先生：その対策方法を、次回説明しましょう。 To be continued next week, same channel, same time!

なな：え？

先生：次回を　こうご期待ということ。

XMLHttpRequest を使って非同期通信で外部サービスを呼び出す方法を「Ajax」（エイジャックス、アジャックス）と呼びます。Ajax は、「Asynchronous JavaScript ＋ XML」の略語。2005年にJesse James Garrett 氏が命名。

第3回　マッシュアップの原理（解決方法）

先生：「同一生成元ポリシー」による制限を回避する対策を説明するわね。ななちゃん、まず、下記のプログラムを実行してみてね。たとえば、test1.html というファイルに書き込んで、ファイルアイコンをダブルクリック。

test1.html

```
<script>
function callbackf(message) {
    document.write(message);
}
</script>
<script>
callbackf("hello");
</script>
```

なな：ブラウザ画面に「hello」と表示されるわ。<script> が二組あるのね。

先生：<script> は、複数あってかまいません。上から順に読み込まれます。次に、下記のプログラムを試してね。

test2.html

```
<script>
function callbackf(message) {
    document.write(message);
}
</script>
<script src=sub.js></script>
```

sub.js

```
callbackf("hello");
```

なな：test2.html のファイルアイコンをダブルクリックすると、さっきと同じように、ブラウザ画面に「hello」と表示されるわね。

先生：sub.js ファイルの内容が読み込まれるわけ。じゃあ、次のは？

test3.html

```
<script>
function callbackf(message) {
    document.write(JSON.stringify(message));
}
</script>
<script src=
"http://geoapi.heartrails.com/api/json?method=searchByPostal&postal=2520816&jsonp=callbackf">
</script>
```

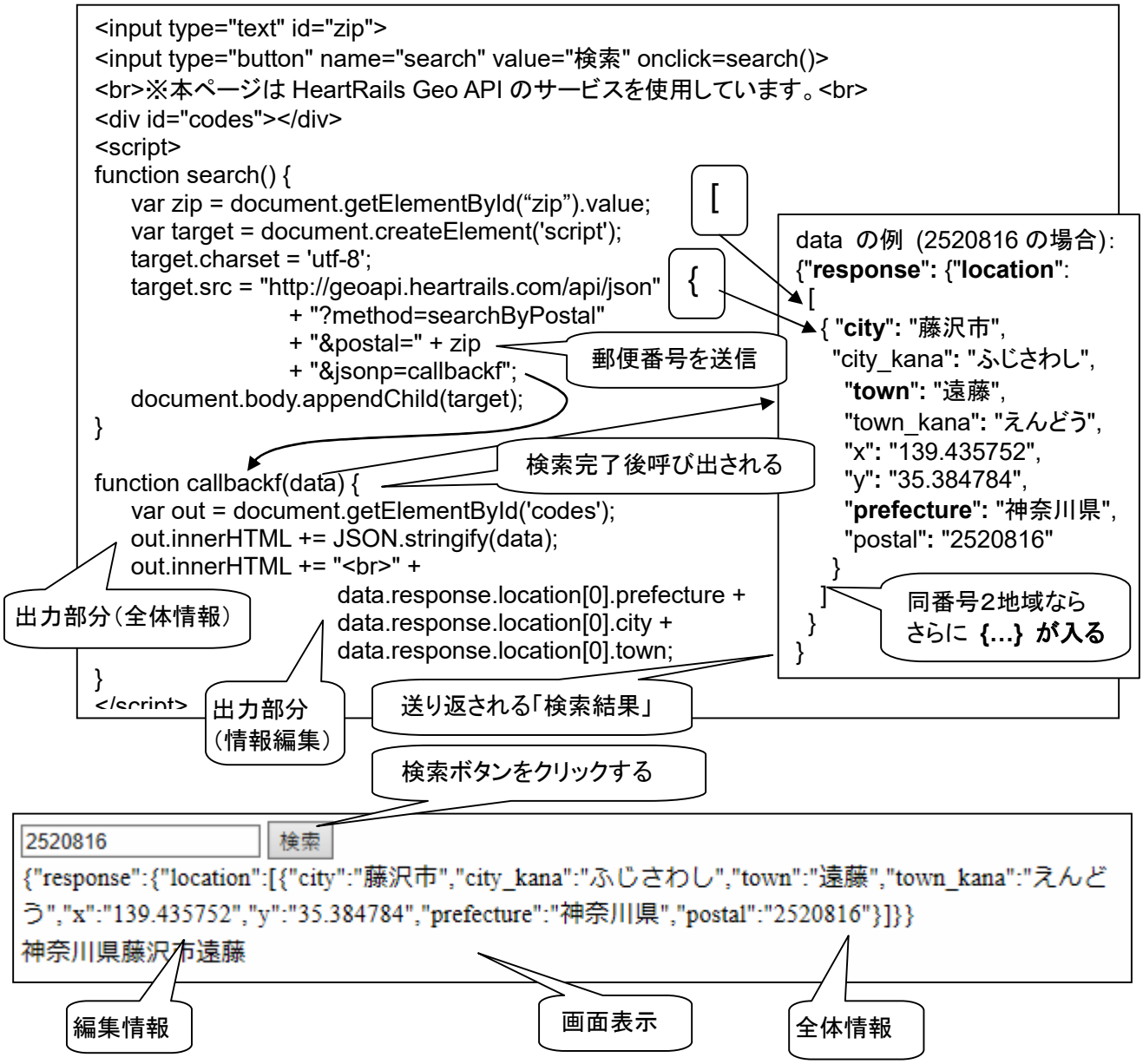
なな：画面に、「{"response":{"location":[{"city":"藤沢市","city_kana":"ふじさわし","town":"遠藤",
"town_kana":"えんどう","x":"139.435752","y":"35.384784","prefecture":"神奈川県",
"postal":"2520816"}]}}」と表示されたわ！

先生：郵便番号「2520816」に対する検索結果が表示されたわね。JSON.stringify() は、複雑なデータを画面に表示するためのものです。

なな：これで完成！？

先生：これだと、郵便番号が変えられないという問題があるの。郵便番号を JavaScript で指定できるようにするには、<script> タグを、JavaScript で組み立てる必要があるの。こういうのを「動的生成」といいます。

test4.html



先生：このような動的生成の方法を「JSONP(*)」とか、「RESTフル（RESTful (**)）インターフェイス」といいます。

なな：とにかく、これで、郵便番号を変えて、検索を依頼できるということね。これって、「同一生成元ポリシー」を破ることにならないの？

先生：破っています（きっぱり！）。この回避策は、セキュリティ強化機構を無視する方法です。オレオレ詐欺にひっかかって、銀行ATMから現金を振り込もうとする時、銀行員に注意されたのに、「いや、私はだまされていない」と言って銀行員を追い払うようなものね。「同一生成元ポリシーは厳しすぎて仕事にならないから無視する」ということ。この範囲なら、プログラマ自身に悪意がなければ問題はない、悪意のある人（犯人、クラッカー）にサイトが乗っ取られたら、もともと、「なんでもあれ」の状態になるだろうということ。現実的に、JSONPはごく一般的な手法として、世の中で広く利用されています。

なな：「赤信号、みんなで渡れば怖くない」かあ。

---

* JSON with Padding。次回説明する JSON（JavaScript Object Notation）形式のデータに callback(…) などを書き加えたものを送り返すので、「詰め物（padding）付きの JSON形式」と呼ばれる。

**: REpresentational State Transfer。他サーバのアプリを利用するインタフェース。World Wide Web創始者の一人、Roy T. Fielding氏が2000年に著した博士論文で提案した。

第4回　マッシュアップの準備（JSON形式）

先生：外部サービスが送り返してくるデータは、ひとつのデータであるより、複数データであることが多いわね。たとえば、郵便番号検索の場合、「神奈川県藤沢市遠藤」というようなデータより、「県名＝神奈川県、市名＝藤沢市、町名＝遠藤」のような複合データのほうが利用しやすいわ。このように、複数のデータをひとまとめにする方法に、XML形式やJSON形式などがあります。

なな：XML形式って何？

先生：HTMLのような Markup Language の1つなの。HTML同様に、World Wide Web Consortium (W3C) により策定されています。eXtensible Markup Language の頭文字で、ユーザがタグ群を定義することによって要素を「拡張」することができるので、こういう名前になっています。このため、メタ言語と呼ばれることもあるの。基本的には「<タグ名>」で要素の始まりを示し、「</タグ名>」で要素の終わりを示す構造をしています。現在では、XHTML、KML、BML 等、多くのフォーマットが XML ベースで開発されています。

なな：じゃあ、JSON は？ ホラー映画の主人公？

先生：XML は柔軟にできていて、多くのデータがこの形式で規定されるようになりました。でも、ファイルサイズが大きくなる、可読性が悪いなどの批判も多くあります。そこで、JavaScript と親和性のよい JSON (JavaScript Object Notation) というデータ記述方法が登場しました。 JSON は、JavaScript のオブジェクトリテラルをベースとしたデータ記述方法です。基本的には、JavaScript のオブジェクトリテラルを踏襲しているが、数値は10進法表記でなければならない、キーの文字列は半角英数字でも「"」で囲まなければならないことになっています。以下にJSONで表現されたデータの例を示します。この例では見やすいように改行を入れてあるけど、改行は必ずしも必要ではありません。JSON はオブジェクトリテラルの一形式と言えます。

```
{
    "name": {
        "familyName": "慶應",
        "firstName": "太郎"
    },
    "age": 19,
    "sex": "male";
}
```

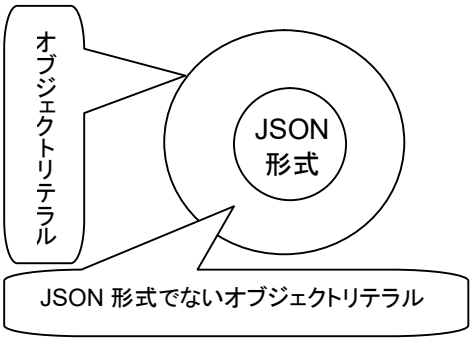
↑JSON形式のデータ

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
  <name>
    <familyName>慶應</familyName>
    <firstName>太郎</firstName>
  </name>
  <age>19</age>
  <sex>male</sex>
</person>
```

↑XML形式のデータ→

```
{
  name: {
      familyName: "慶應",
      firstName: "太郎"
  },
  age: 19,
  sex: "male"
};
```

「オブジェクトリテラル」の例

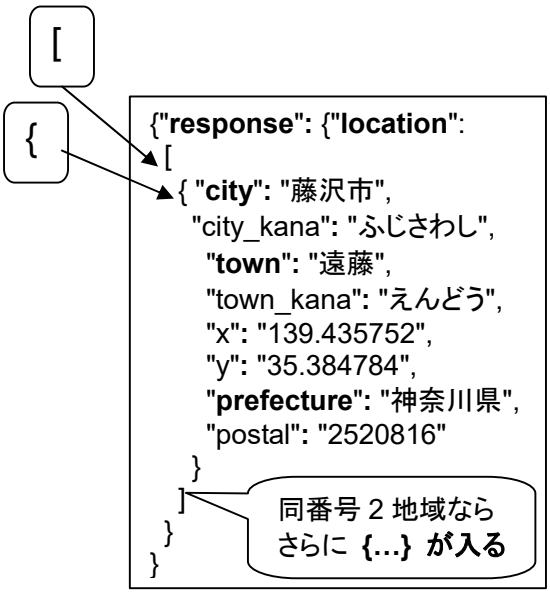


JSON形式は、オブジェクトリテラルの一種、特別な形。たとえれば、JSON形式はコイン、オブジェクトリテラルは貨幣。

先生：郵便番号の検索結果の例で説明してみましょう。

{"response":{"location":[{"city":"藤沢市","city_kana":"ふじさわし","town":"遠藤",
"town_kana":"えんどう","x":"139.435752","y":"35.384784","prefecture":"神奈川県",
"postal":"2520816"}]}}

見やすくするために、改行を追加してみましょう。



このデータを data という変数に記憶させた場合、県名、市名、町名を取り出してつなぐには下記のようにします。

```
var s = data.response.location[0].prefecture +
        data.response.location[0].city +
        data.response.location[0].town;
```

結果は、「神奈川県藤沢市遠藤」になります。緯度は「data.response.y.」、経度は「data.response.x.」で取り出すことができます。

なな：りょーかい。

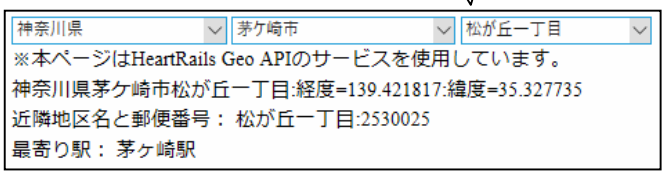
第5回　さあ、いよいよマッシュアップ

先生：heartrails geoapiの「県名/市名/町名選択コンボボックス => 緯度/経度」と、「緯度/経度 -> 近隣地区と郵便番号」と、「郵便番号 => 最寄り駅」をつなぐ（マッシュアップする）例です。

```
      <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
      <script src="http://geoapi.heartrails.com/api/geoapi.js"></script>
      <select id="geoapi-prefectures" name="geoapi-prefectures">
        <option value="都道府県を選択してください">都道府県を選択してください</option>
      </select>
      <select id="geoapi-cities" name="geoapi-cities">
        <option value="市区町村名を選択してください">市区町村名を選択してください</option>
      </select>
      <select id="geoapi-towns" name="geoapi-towns"
                onChange='myfunc(geoapi_towns[this.selectedIndex-1])'>
        <option value="町域を選択してください">町域を選択してください</option>
      </select>
      <br>※本ページは HeartRails Geo API のサービスを使用しています。<br>
      <div id=out></div>
      <script>
var outp = document.getElementById("out");
// -------   heartrails geoapi 県名/市名/町名選択コンボボックス => 緯度/経度 ---------
function myfunc(data) {
    outp.innerHTML = data.prefecture + data.city + data.town
                    + ":経度=" + data.x + ":緯度=" + data.y + "<br>";
    findZip(data.y, data.x);
}
// ------ マッシュアップ(1) heartrails geoapi 緯度/経度 -> 近隣地区と郵便番号 --------
function findZip(lat,lon) {
    var target = document.createElement('script');
    target.charset = 'utf-8';
    target.src = "http://geoapi.heartrails.com/api/json"
                    + "?method=searchByGeoLocation"
                    + "&y=" + lat
                    + "&x=" + lon
                    + "&jsonp=callbackfn";
    document.body.appendChild(target);
}

function callbackfn(data) {
    var o = "近隣地区名と郵便番号： ";
    o += data.response.location[0].town+":"+data.response.location[0].postal+"<br>";
    outp.innerHTML += o;
findStation(data.response.location[0].postal);
}
// ------------- マッシュアップ(2) heartrails geoapi 郵便番号 => 最寄り駅 ---------------
function findStation(zip) {
    var target = document.createElement('script');
    target.charset = 'utf-8';
    target.src = "http://geoapi.heartrails.com/api/json"
                    + "?method=getStations"
                    + "&postal=" + zip
                    + "&jsonp=callbackfn2";
    document.body.appendChild(target);
}

function callbackfn2(data) {
    var o = "最寄り駅： ";
    for (var i in data.response.station)
        o += data.response.station[i].name+"駅<br>";
    outp.innerHTML += o;
}
      </script>
```

次のサービスの呼び出し

実行結果の例



次のサービスの呼び出し

第6回　マニュアルの見方

http://geoapi.heartrails.com/api.html （サービス提供者の API 説明ページ）

「HeartRails Geo API」 は、郵便番号／住所／緯度経度データ等の地理情報を、XML、JSON(P) 形式の API により無料でご提供させていただくサービスです。この API をご利用になることにより、お客様側ではサーバーサイドの処理を実装することなく、サンプルのようなアプリケーションを簡単に開発することができます。
（ジオコーディング、逆ジオコーディングにも対応しております。）

API （XML／JSON）　サンプル　API　データ販売

API （XML／JSON）

API は商用、非商用を問わず、無料でご利用になれます。
ご利用条件の詳細に関しましては、位置参照情報利用約款、HeartRails の利用規約、および免責事項をご覧ください。

無料でご利用になられる際には、アプリケーション内に 「HeartRails Geo API」 のクレジットを記載してください。
大規模サイトでご利用になられる際には、有料プランもご検討ください。

1

次ページ参照

API の一覧

- エリア情報取得 API
- 都道府県情報取得 API
- 市区町村情報取得 API
- 町域情報取得 API
- 最寄駅情報取得 API
- 郵便番号による住所検索 API
- 緯度経度による住所検索 API
- キーワードによる住所検索 API
- 「エリア名」「市区町村名」「町域名」の連結コンボボックス
- 「都道府県名」「市区町村名」「町域名」の連結コンボボックス
- 「郵便番号」による住所検索フォーム

「HeartRails Geo API」 の API は、全て GET 形式のリクエスト、UTF-8 形式の入力、出力に対応しております。

http://geoapi.heartrails.com/api.html#postal

郵便番号による住所検索 API

ご指定の郵便番号に合致する町域の情報の一覧を取得する API です。

リクエスト URL

フォーマット	URL
XML 形式	http://geoapi.heartrails.com/api/xml?method=searchByPostal
JSON(P) 形式	http://geoapi.heartrails.com/api/json?method=searchByPostal

リクエストパラメータ

パラメータ	値	説明
method	searchByPostal （固定）	メソッド名
postal	string （必須）	URL エンコード （UTF-8） された郵便番号 7 桁の半角数字を入力してください
jsonp	string （オプション）	JSON 形式のデータを受け取るためのコールバック関数名 JSON 形式のリクエスト URL にのみ対応

レスポンスフィールド

フィールド	説明
response	町域の情報の一覧
prefecture	町域の存在する都道府県名
city	町域の存在する市区町村名
city-kana	町域の存在する市区町村名の読み仮名 （平仮名）
town	町域名
town-kana	町域名の読み仮名 （平仮名）
x	町域の経度 （世界測地系）
y	町域の緯度 （世界測地系）

```
<input type="text" id="zip">
<input type="button" name="search" value="検索" onclick=search()>
<br>※本ページは HeartRails Geo API のサービスを使用しています。<br>
<div id="codes"></div>
<script>
function search() {
    var zip = document.getElementById(“zip”).value;
    var target = document.createElement('script');
    target.charset = 'utf-8';
    target.src = "http://geoapi.heartrails.com/api/json"
                    + "?method=searchByPostal"
                    + "&postal=" + zip
                    + "&jsonp=callbackf";
    document.body.appendChild(target);
}
function callbackf(data) {
    var out = document.getElementById('codes');
    out.innerHTML += JSON.stringify(data);
    out.innerHTML += "<br>" +
                    data.response.location[0].prefecture +
                    data.response.location[0].city +
                    data.response.location[0].town;
}
</script>
```

上記の
マニュアル箇所
に対応

実際に送られてきたデータを JSON.stringify() で表示して確認

第7回　使用上の注意

Webサービスを利用するにあたってはいくつかの注意が必要です。 ここでは、それらの注意点についてみてゆきましょう。

<プログラミング>

Webサービスを使うということは、他人のサイトにアクセスすることに他ならない。このため、無限ループの中でWebサービスを使っていたりすると、他人のサイトに高速に大量のアクセスをすることになる。たとえプログラムを書いた本人に悪意はなくとも、このような行為はWebサービスを提供している側から見るとDoS攻撃 (Denial of Service attack) と見えてしまう。Webサービスを利用する場合は、Webサービスを提供している側に迷惑をかけないよう、細心の注意を払ってプログラムを作成するように心がける必要がある。

<セキュリティ>

自分のウェブページに他者のWebサービスを埋め込む場合は、 その品質を十分に吟味する必要がある。 悪意を持った他人のWebサービスを埋め込んでしまうと、悪事に加担したことになり、共犯者になってしまう可能性がある。

たとえば、見ず知らずの他人から教えてもらったWebサービスを安易に利用してしまうと、表向きは便利な機能でも、そのWebサービスが個人情報を流出するなど悪意を持った動作をするかもしれない。

このようなトラブルを回避するには、多くの人が長期に渡って利用しているなどの安全性が十分確認されているようなWebサービスを選ぶと良い。

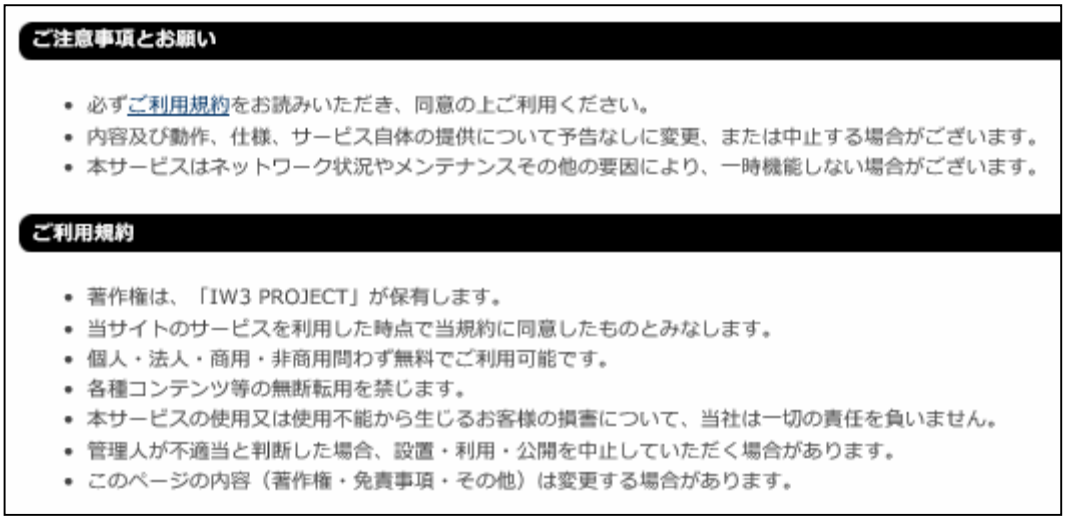
<使用許諾条件>

Webサービスを利用する場合、「使用許諾条件」を確認し、それを守る必要がある。たとえば、提供元を明示したり、特定のアイコンを表示することを求められている場合も多くある。場合によってはユーザ登録が必要なこともある。商用利用（企業ウェブサイトでの利用や、有料のサービス提供への利用）を禁止している場合もある。

例えば、友人のホームページのソースの一部をコピー（まね）して利用する場合などに、特に注意する必要がある。サービス呼び出し部分だけでなく、クレジット部分もコピーする必要があるかも知れないし、URLが変ることになるので改めてユーザ登録を行う必要がある場合もある。対象Webサービス提供者のホームページなどで、使用許諾条件を確認することが重要である。

付録1　外部サービス:カレンダー

http://project.iw3.org/jCalendar/



```
<style>
    table { border-collapse:collapse; }
    td,th { border: solid 1px; width: 100px; text-align: center }
</style>
西暦年月（6桁半角数字、200001～203812）:
<input type="text" name="idata" id="idata" value="">
<input type="button" name="search" value="実行" onclick=search()>
<p><div id="codes"></div>
<script>
var zip_data = null;

function search() {
      var idata = document.getElementById('idata').value;
      var target = document.createElement('script');
      target.charset = 'utf-8';
      target.src = "http://api.thni.net/jCalendar/jData/jsonp/" + idata + ".js" + "?jsoncallback=?";
      document.body.appendChild(target);
}

function InitCal(data) {
      var out = document.getElementById('codes');
      var t, o = "<table>" + "<tr><th>日</th><th>月</th><th>火</th>"
                                  +       "<th>水</th><th>木</th><th>金</th><th>土</th></tr>";
      // out.innerHTML += JSON.stringify(data);
      for (var i in data) {
          if (i%7==0) o += "<tr>"
          o += "<td>" + (((t=data[i].jDay)       ==="")?"  ":t)
             +   "<br>" + (((t=data[i].jHoliday)==="")?"  ":t) + "</td>";
          if (i%7==6) o += "</tr>";
      }
      o += "</table>"; out.innerHTML = o; //alert(o);
}
</script>
```

西暦年月（６桁半角数字、200001～203812）：201705 実行

日	月	火	水	木	金	土
	1	2	3 憲法記念日	4 みどりの日	5 こどもの日	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

付録2　外部サービス:天気予報

http://www.drk7.jp/weather/

drk7.jp

パパ料理のレシピやガジェット情報を紹介

料理レシピ　日々の戯言　ガジェットや家電　drk7jp-Webサービス　Perl言語　MovableType　ネットサービス
データベース技術　ソフトウェア　Installノウハウ　ゲーム関連　その他レビュー等

気象庁の天気予報情報を XML で配信

Japan Weather Forecast xml （日本お天気予報）の配信リスト

気象庁が公開している天気予報情報（http://www.jma.go.jp/）から独自の手法でメタデータ化した情報を XML 形式で配信しております。配信リストは下記の一覧の通りです。

Japan Weather Forecast json （日本お天気予報）の配信リスト

上記のメタデータを JSONP 形式で配信しております。callback関数名は drk7jpweather.callback() になります。配信リストは下記の一覧の通りです。

北海道	：北海道
東北	：青森県 / 岩手県 / 宮城県 / 秋田県 / 山形県 / 福島県
関東	：東京都 / 神奈川県 / 埼玉県 / 千葉県 / 茨城県 / 栃木県 / 群馬県 / 山梨県
信越	：新潟県 / 長野県
北陸	：富山県 / 石川県 / 福井県
東海	：愛知県 / 岐阜県 / 静岡県 / 三重県
近畿	：大阪府 / 兵庫県 / 京都府 / 滋賀県 / 奈良県 / 和歌山県
中国	：鳥取県 / 島根県 / 岡山県 / 広島県 / 山口県
四国	：徳島県 / 香川県 / 愛媛県 / 高知県
九州	：福岡県 / 佐賀県 / 長崎県 / 熊本県 / 大分県 / 宮崎県 / 鹿児島県
沖縄	：沖縄県

ご利用上の注意点　※ご利用前に必ずご一読ください。

Japan Weather Forecast XML のデータは気象庁が公開している天気予報情報（http://www.jma.go.jp/）のみから独自の手法でメタデータ化して XML 配信するサービスです。本サービスは税金で調査された天気予報は、その他の国が提供するデータ同様に私的な目的で利用可能という考えのもと、実験的な意味合い公開しています。またここで公開する天気予報 XML は当サイトの MTWeather などのように私的な目的で利用することを前提としたものです。商用利用等の個人的利用以外の目的でのご利用はお控えください。

XML のフォーマットは暫定的なものです。今後のバージョンアップにて予期せずフォーマットが変更される可能性があります。また、ご要望等ございましたら遠慮なくお問い合わせフォームからご意見ください。

本サービスを利用することにより発生する損害については一切保障しません。また何らかの問題が発生した場合、予告なく早期にサービス停止いたします。

```
都道府県を選択:
<select name="pref" onChange="go(this[this.selectedIndex].value)">
      <option value="0" selected>-----
      <option value="01">北海道<option value="02">青森県<option value="03">岩手県
      <option value="04">宮城県<option value="05">秋田県<option value="06">山形県
      <option value="07">福島県<option value="08">茨城県<option value="09">栃木県
      <option value="10">群馬県<option value="11">埼玉県<option value="12">千葉県
      <option value="13">東京都<option value="14">神奈川県<option value="15">新潟県
      <option value="16">富山県<option value="17">石川県<option value="18">福井県
      <option value="19">山梨県<option value="20">長野県<option value="21">岐阜県
      <option value="22">静岡県<option value="23">愛知県<option value="24">三重県
      <option value="25">滋賀県<option value="26">京都府<option value="27">大阪府
      <option value="28">兵庫県<option value="29">奈良県<option value="30">和歌山県
      <option value="31">鳥取県<option value="32">島根県<option value="33">岡山県
      <option value="34">広島県<option value="35">山口県<option value="36">徳島県
      <option value="37">香川県<option value="38">愛媛県<option value="39">高知県
      <option value="40">福岡県<option value="41">佐賀県<option value="42">長崎県
      <option value="43">熊本県<option value="44">大分県<option value="45">宮崎県
      <option value="46">鹿児島県<option value="47">沖縄県
</select><br><div id=out></div>
<script>
function go(n) {
    var target = document.createElement('script');
    target.src = "http://www.drk7.jp/weather/json/" + n + ".js";
    document.body.appendChild(target);
}

var drk7jpweather = { callback: function(data) {
    var out = document.getElementById('out');
    out.innerHTML = ""; // out.innerHTML += JSON.stringify(data.pref.area);
    for (var i in data.pref.area) {
        out.innerHTML += i + "<br>"
        for (var j in data.pref.area[i].info) {
            out.innerHTML += (" " + data.pref.area[i].info[j].date+":"
                              + data.pref.area[i].info[j].weather + "<br>");
}  }  }  }
</script>
```

付録3　外部サービス：国土地理院の標高API

http://maps.gsi.go.jp/development/api.html　　緯度、経度から、標高を得る

標高API

2013年3月14日より、指定した位置の標高値を取得するウェブAPIを試験公開しています。
ユーザ独自の地図表示サイトやプログラムから本APIにアクセスし、標高値を取得して表示することができます。
ご利用の際には、使用上の注意をご確認ください。

使用方法

以下の形式パラメータを指定することで標高値を得ることができます。

http://cyberjapandata2.gsi.go.jp/general/dem/scripts/getelevation.php?<パラメータ>

インプットパラメータ	意味	備考	パラメータの指定について
lon	経度	度の10進法で指定します	必ず指定してください
lat	緯度	度の10進法で指定します	必ず指定してください
callback	JSONPで返すときのコールバック関数	"_(アンダースコア)"、"$"以外の記号及びJavaScriptの予約語を使用することはできません	どちらか一方を指定してください
outtype	アウトプットの形式	「JSON」という固定文字列を指定します	

「callback」を指定するとJSONP形式で、「outtype」を指定するとJSON形式で結果が返ります。
「callback」と「outtype」はどちらかを指定してください。（両方の指定はしないでください）

アウトプットパラメータ	意味	備考
elevation	標高値	エラーの場合は「-----」という文字列を返します
hsrc	標高データのデータソース	エラーの場合は「-----」という文字列を返します

「hsrc」は「5m（レーザ）」「5m（写真測量）」「10m」のいずれかの値を取ります。

「hsrc」の値とその意味、標高値の桁数の関係は以下の表のようになっています。
上のものほど計測精度が良いです。本APIでは、その地点における最も精度の良いhsrcでの値を返しています。
（※参考：各データソースのより詳しい仕様や精度 [PDF 574KB]）

hsrc	意味	「elevation」の桁数
5m（レーザ）	標高値の元データが航空レーザー測量で得られた5m DEMであることを意味しています。	0.1mの位まで
5m（写真測量）	標高値の元データが写真測量で得られた5m DEMであることを意味しています。	
10m	標高値の元データが等高線から得られた10m DEMであることを意味しています。	1mの位まで

使用例

戻り値をJSONP形式で得たい場合

http://cyberjapandata2.gsi.go.jp/general/dem/scripts/getelevation.php?lon=140.08531&lat=36.103543&callback=myfunc

戻り値をJSON形式で得たい場合

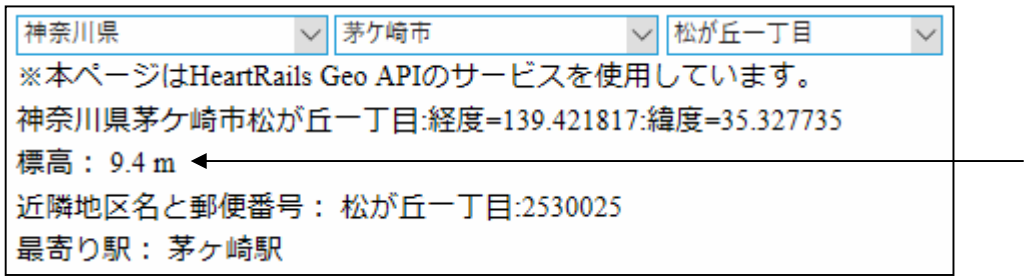
http://cyberjapandata2.gsi.go.jp/general/dem/scripts/getelevation.php?lon=140.08531&lat=36.103543&outtype=JSON

使用上の注意

- 地理院標高APIを提供するサーバに過度の負担を与えないでください。過度の負担を与えると判断したアクセスについて、国土地理院は予告なく遮断を行う場合があります。
- 国土地理院は、地理院標高APIを予告なく内容変更したり提供停止したりする可能性があります。

第5回のプログラム末尾に追加

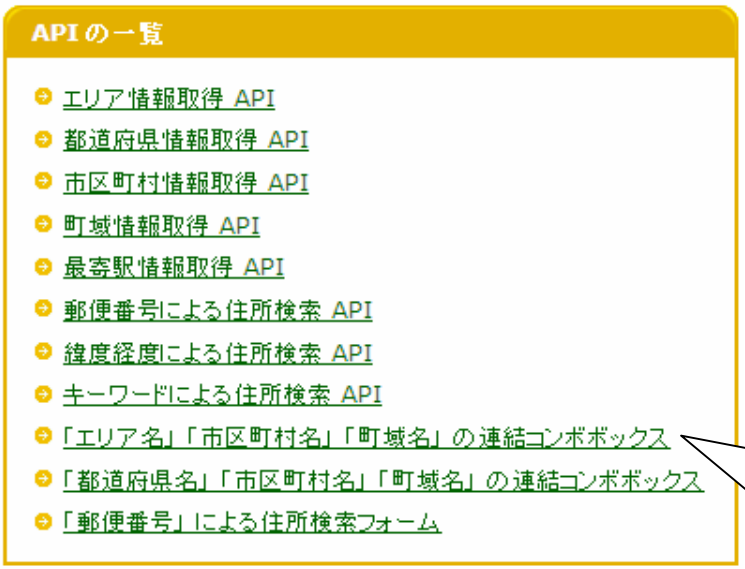
```
// ---------------------------------------------
function myfunc(data) {
    outp.innerHTML = data.prefecture + data.city + data.town
                        + ":経度=" + data.x + ":緯度=" + data.y + "<br>";
    findZip(data.y, data.x);
    findHeight(data.y, data.x)
}
// ------  マッシュアップ(1) heartrails geoapi  緯度/経度  ->  近隣地区と郵便番号  --------
function findZip(lat,lon) {
:
// ------  マッシュアップ(3)  国土地理院  標高情報：緯度/経度  ->  標高  --------
function findHeight(lat,lon) {//alert("lat/lon:"+lat+", lon:"+lon);
    var target = document.createElement('script');
    target.src = "http://cyberjapandata2.gsi.go.jp/general/dem/scripts/getelevation.php"
                    + "?lon=" + lon
                    + "&lat=" + lat
                    + "&callback=callbackfn3";
    document.body.appendChild(target);
}

function callbackfn3(data) {
    var o = "標高：  ";
    o += data.elevation +" m<br>";
    outp.innerHTML += o;
}
```



実行例　　非同期動作なので、標高は末尾に表示される場合もある。

付録4　外部サービス：HeartRails Geo API

http://geoapi.heartrails.com/api.html



コンボ(combo)は、コンビネーション（combination）の略語で、集合体、組み合わせなどの意味を持つ英語。

名称	method	入力	出力
エリア情報取得 API	getAreas		エリア名
都道府県情報取得 API	getPrefectures	エリア名	都道府県名
市区町村情報取得 API	getCities	エリア名‖都道府県名	市区町村名
町域情報取得 API	getTowns	都道府県名‖市区町村名	都道府県名/市区町村名/町域名/経度/緯度/郵便番号
最寄駅情報取得 API	getStations	郵便番号	最寄駅名/前の駅名/次の駅名/経度/緯度/距離/郵便番号/都道府県名/路線名
郵便番号による住所検索 API	searchByPostal	郵便番号	都道府県名/市区町村名/町域名/経度/緯度/郵便番号
緯度経度による住所検索 API	searchByGeoLocation	**経度/緯度**	都道府県名/市区町村名/町域名/経度/緯度/郵便番号
キーワードによる住所検索 API	suggest	キーワード/前方一致‖部分一致‖後方一致	都道府県名/市区町村名/町域名/経度/緯度/郵便番号
「エリア名」「市区町村名」「町域名」の連結コンボボックス			
「都道府県名」「市区町村名」「町域名」の連結コンボボックス			
「郵便番号」による住所検索フォーム		「郵便番号」を入力すると住所を検索して補完するフォーム	

「キーワードによる住所検索」は、地名に関するキーワードによる検索。施設名からの検索ではない。

「コンボボックス」は、結構　かっこいい。コンボはコンビネーションの略で、「連結」「組み合わせ」という意味。

HeartRails Geo API　市区町村情報取得API

```
データ
<input type="text" name="idata" id="idata" value="">
<input type="button" name="search" value="実行" onclick=search()><br>
<div id="codes"></div>
<script>
function search() {
   var idata = document.getElementById('idata').value;
   var target = document.createElement('script');
   target.charset = 'utf-8';
   target.src = "http://geoapi.heartrails.com/api/json"
                + "?method=getCities"
                + "&prefecture=" + encodeURIComponent(idata)
                + "&jsonp=callbackf";
   document.body.appendChild(target);
}

function callbackf(data) {
   var out = document.getElementById('codes');
   out.innerHTML += JSON.stringify(data);
}
</script>
```

市区町村情報取得 API

リクエスト URL

フォーマット	URL
JSON(P) 形式	http://geoapi.heartrails.com/api/json?method=getCities

リクエストパラメータ

パラメータ	値	説明
method	getCities （固定）	メソッド名
prefecture	string	URL エンコード （UTF-8） された都道府県名

データ 石川県 実行

{"response":{"location":[{"city":"金沢市","city_kana":"かなざわし"},{"city":"七尾市","city_kana":"ななおし"},{"city":"小松市","city_kana":"こまつし"},{"city":"輪島市","city_kana":"わじまし"},{"city":"珠洲市","city_kana":"すずし"},{"city":"加賀市","city_kana":"かがし"},{"city":"羽咋市","city_kana":"はくいし"},{"city":"かほく市","city_kana":"かほくし"},{"city":"白山市","city_kana":"はくさんし"},{"city":"能美市","city_kana":"のみし"},{"city":"川北町","city_kana":"かわきたまち"},{"city":"川北町","city_kana":"のみぐんかわきたまち"},{"city":"野々 市町","city_kana":"ののいちまち"},{"city":"野々 市町","city_kana":"いしかわぐんののいちまち"},{"city":"津幡町","city_kana":"かほくぐんつばたまち"},{"city":"津幡町","city_kana":"つばたまち"},{"city":"内灘町","city_kana":"かほくぐんうちなだまち"},{"city":"内灘町","city_kana":"うちなだまち"},{"city":"志賀町","city_kana":"はくいぐんしかまち"},{"city":"志賀町","city_kana":"しかまち"},{"city":"宝達志水町","city_kana":"はくいぐんほうだつしみずちょう"},{"city":"宝達志水町","city_kana":"ほうだつしみずちょう"},{"city":"中能登町","city_kana":"かしまぐんなかのとまち"},{"city":"中能登町","city_kana":"なかのとまち"},{"city":"穴水町","city_kana":"ほうすぐんあなみずまち"},{"city":"穴水町","city_kana":"あなみずまち"},{"city":"能登町","city_kana":"のとちょう"},{"city":"能登町","city_kana":"ほうすぐんのとちょう"}]}}

付録5　外部サービス：駅データ. jp

http://www.ekidata.jp/api/

・日本の鉄道駅の情報（駅データ）を配布するサイト。
・当サイトで提供している駅データは、法人・個人及び、商用・非商用を問わず、誰でも利用できる。詳細は「利用規約」(http://www.ekidata.jp/agreement.php)。
・当サイトの駅データを用いて生じたいかなる損害についても弊社は責任を負うことはできない。

目的
・現在、多くのWEBサイトで駅情報が利用されているが、異なるサイトでは同じ駅でありながらそのデータ仕様が異なっている。駅データの仕様を統一化することで、駅データを用いている異なるサービス同士のデータ連携が行いやすくなり、ユーザーへのサービス向上に結びつく。 『駅データ.jp』は、無料で駅データを公開することで、駅データの仕様を共有し、全国の駅情報を利用したサービスの活性化を目指している。

運用
・『駅データ.jp』では、データの更新申請をサイト上で受け付けている。
・情報調査は行っておりますが、データの正確性をより高めるために、データに誤りを見つけた場合や、駅情報の更新があった際には情報提供に協力を。

名称	code	入力	出力
都道府県 API	p	都道府県コード	沿線コード（路線コード）/沿線名
路線 API	l	路線コード	駅コード/駅グループコード/駅名/経度/緯度
駅詳細 API	s	駅コード	都道府県コード/沿線コード/沿線名/駅コード/駅グループコード/駅名/**経度/緯度**
駅グループ API	g	駅コード	グループ駅_都道府県コード/沿線コード/沿線名/駅コード/駅名
隣接駅 API	n	路線コード	1 つめの駅コード/2 つめの駅コード/1 つめの駅名/2 つめの駅名/1 つめの経度/1 つめの緯度/2 つめの経度/2 つめの緯度

「都道府県→沿線→駅」のメニュー選択プログラムがある。メニュー操作で駅コードを得ることができる。

駅データ. jp 都道府県API

・都道府県の路線一覧を表示する JSON データ。
・北海道から沖縄まで47都道府県の沿線を表示できる。
・都道府県APIに新幹線データは含まれていない。
・JSON形式API ： http://www.ekidata.jp/api/p/(都道府県コード).json
・都道府県コードは、1～47の半角数字。
・都道府県コード:
1 北海道 2 青森県 3 岩手県 4 宮城県 5 秋田県 6 山形県 7 福島県 8 茨城県
9 栃木県 10 群馬県 11 埼玉県 12 千葉県 13 東京都 14 神奈川県 15 新潟県 16 富山県
17 石川県 18 福井県 19 山梨県 20 長野県 21 岐阜県 22 静岡県 23 愛知県 24 三重県
25 滋賀県 26 京都府 27 大阪府 28 兵庫県 29 奈良県 30 和歌山県 31 鳥取県 32 島根県
33 岡山県 34 広島県 35 山口県 36 徳島県 37 香川県 38 愛媛県 39 高知県 40 福岡県
41 佐賀県 42 長崎県 43 熊本県 44 大分県 45 宮崎県 46 鹿児島県 47 沖縄県 99 その他

路線API

・路線の駅一覧を表示する JSON データ。
・路線コードから、該当する**駅一覧**を表示できる。
・路線APIに新幹線データは含まれていない。
・JSON形式API: http://www.ekidata.jp/api/l/(路線コード).json
・路線コードは5桁数値。

駅詳細API

・駅の詳細を表示する JSON データ。
・駅コードから、該当する駅の詳細を表示できる。
・駅詳細APIに新幹線データは含まれていない。
・JSON形式API ： http://www.ekidata.jp/api/s/(駅コード).json
・駅コードは7桁数値。

駅グループAPI

・駅グループ一覧を表示する JSON データ。
・駅コードから、同一駅の他の路線駅一覧を表示できる。
・駅グループAPIに新幹線データは含まれていない。
・JSON形式API ： http://www.ekidata.jp/api/g/(駅コード).json
・都道府県コードは、1～47の半角数字。

隣接駅API

・路線の隣接駅一覧を表示する JSON データ。
・路線コードに含まれる隣接駅をすべて表示する。
・隣接駅APIに新幹線データは含まれていない。
・JSON 形式 API: http://www.ekidata.jp/api/n/(路線コード).json

都道府県→沿線→駅

・都道府県を選択すると、該当する沿線を表示する。
・さらに、沿線を選択すると、該当する駅を表示する。
・サンプルプログラム:http://www.ekidata.jp/api/sample_drilldown.php

利用条件
■ 利用の制限
商用、非商用にかかわらず、どなたでもご利用いただくことができますが、1日に10万アクセスを超えるご利用の場合は、あらかじめ弊社にご連絡ください。
また、短時間に大量のアクセスを行わないようにしてください。
■ 提供の変更及び終了
データ提供サービスについては予告なく変更・サービスを終了する場合があります、ご了承下さい。
■ 免責事項
データ提供サービスをご利用することにより生じたいかなる損害についても弊社は責任を負いません。
■ 禁止事項
データ提供サービスの以下の利用を禁じます。
(1) 犯罪を勧誘又は助長するおそれのある利用
(2) わいせつ、裸体、死体、その他公序良俗に反する情報に関係する利用
(3) 消費者の判断に錯誤を与える恐れのある利用
(4) 第三者の財産権(知的所有権を含む)の侵害、プライバシーの侵害、誹謗中傷その他の不利益を与える利用
(5) 弊社または本件サービスへの誹謗中傷、弊社業務の運営・維持を妨げる利用
(6) ウイルスその他第三者に害悪を加えるおそれのある利用

使用例：都道府県API　の、神奈川県（コード 14）のデータ全体（全路線）と、最初のデータを表示

```
<script src="http://www.ekidata.jp/api/p/14.json"></script>
<button onclick="getJSON()">Get JSON</button>
<script>
function getJSON() {
   alert(JSON.stringify(xml.data));    alert(xml.data.line[0].line_name); }
</script>
```

↑ボタンをクリックすると、全データ、次に「JR東海道本線(東京～熱海)」と alert 表示する。

http://www.ekidata.jp/api/p/14.json　が返す内容

xml.data に
結果を代入

```
if(typeof(xml)=='undefined') xml = {};
xml.data =
{"line":[{"line_cd":11301,"line_name":"JR 東海道本線(東京～熱海)"},
{"line_cd":11303,"line_name":"JR 南武線"},
{"line_cd":11304,"line_name":"JR 鶴見線"},
{"line_cd":11306,"line_name":"JR 横浜線"},
{"line_cd":11307,"line_name":"JR 根岸線"},
{"line_cd":11308,"line_name":"JR 横須賀線"},
{"line_cd":11309,"line_name":"JR 相模線"},
{"line_cd":11311,"line_name":"JR 中央本線(東京～塩尻)"},
{"line_cd":11328,"line_name":"JR 成田エクスプレス"},
{"line_cd":11332,"line_name":"JR 京浜東北線"},
{"line_cd":11333,"line_name":"JR 湘南新宿ライン"},
{"line_cd":11505,"line_name":"JR 御殿場線"},
{"line_cd":24002,"line_name":"京王相模原線"},
{"line_cd":25001,"line_name":"小田急線"},
{"line_cd":25002,"line_name":"小田急江ノ島線"},
{"line_cd":25003,"line_name":"小田急多摩線"},
{"line_cd":26001,"line_name":"東急東横線"},
{"line_cd":26002,"line_name":"東急目黒線"},
{"line_cd":26003,"line_name":"東急田園都市線"},
{"line_cd":26004,"line_name":"東急大井町線"},
{"line_cd":26008,"line_name":"東急こどもの国線"},
{"line_cd":27001,"line_name":"京急本線"},
{"line_cd":27003,"line_name":"京急大師線"},
{"line_cd":27004,"line_name":"京急逗子線"},
{"line_cd":27005,"line_name":"京急久里浜線"},
{"line_cd":29001,"line_name":"相鉄本線"},
{"line_cd":29002,"line_name":"相鉄いずみ野線"},
{"line_cd":99310,"line_name":"みなとみらい線"},
{"line_cd":99314,"line_name":"伊豆箱根鉄道大雄山線"},
{"line_cd":99316,"line_name":"ブルーライン"},
{"line_cd":99317,"line_name":"金沢シーサイドライン"},
{"line_cd":99320,"line_name":"江ノ島電鉄線"},
{"line_cd":99326,"line_name":"湘南モノレール"},
{"line_cd":99339,"line_name":"箱根登山鉄道鉄道線"},
{"line_cd":99343,"line_name":"グリーンライン"},
{"line_cd":99344,"line_name":"箱根登山ケーブルカー"}]}
if(typeof(xml.onload)=='function') xml.onload(xml.data);
```

xml.data を引数にして
xml.onload を
呼び出す

↓県名を指定できるようにしたもの。

```
都道府県コード(1～47)<input type="text" name="idata" id="idata" value="14">
<button onclick="getJSON()">Get JSON</button>
<script>
var xml;
function getJSON() {
   xml = {};
   var idata = document.getElementById('idata').value;
   var script = document.createElement('script');
   script.src = "http://www.ekidata.jp/api/p/" + idata + ".json";
   document.body.appendChild(script);
   xml.onload = dply; }
function dply(data) {
   alert(JSON.stringify(data)); alert(data.line[0].line_name); }
</script>
```

■ リクエストURL

JSON形式 ： **http://www.ekidata.jp/api/p/(都道府県コード).json**

■ JSON定義

FIRST TAG	SECOND TAG	内容
line	line_cd	沿線コード
	line_name	沿線名

```
{"line":[
   {"line_cd":11301,"line_name":"JR 東海道本線(東京～熱海)"},
   {"line_cd":11303,"line_name":"JR 南武線"},
   :
```

JR 東海道本線(東京～熱海)

使用例：駅詳細API

```
駅コード<input type="text" name="idata" id="idata" value="1130108">
<button onclick="getJSON()">Get JSON</button>
<script>
var xml;
function getJSON() {
   xml = {};
   var idata = document.getElementById('idata').value;
   var script = document.createElement('script');
   script.src = "http://www.ekidata.jp/api/s/" + idata + ".json";
   document.body.appendChild(script);
   xml.onload = dply; }
function dply(data) {
   alert(JSON.stringify(data));
   alert(data.station[0].station_name
            + "  緯度:" + data.station[0].lon
            + "  経度:" + data.station[0].lat);
}
</script>
```

```
{"station":[{
    "pref_cd":14,"line_cd":11301,
    "line_name":"JR 東海道本線(東京～熱海)",
    "station_cd":1130108,"station_g_cd":1130108,
    "station_name":"藤沢","lon":139.487293,"lat":35.338882}]}
```

藤沢　緯度:139.487293　経度:35.338882

使用例：隣接駅API

東海道線

```
路線コード<input type="text" name="idata" id="idata" value="11301">
<button onclick="getJSON()">Get JSON</button>
<script>
var xml;
function getJSON() {
   xml = {};
   var idata = document.getElementById('idata').value;
   var script = document.createElement('script');
   script.src = "http://www.ekidata.jp/api/n/" + idata + ".json";
   document.body.appendChild(script);
   xml.onload = dply; }
function dply(data) {
   alert(JSON.stringify(data));
   var opt = "" + data.station_join[0].station_name2;
   for(var x=0; x<data.station_join.length; ++x) {
        opt += "/" + data.station_join[x].station_name1;
   }
   alert(opt);
}
</script>
```

熱海/湯河原/真鶴/根府川/早川/小田原/鴨宮/国府津/二宮/大磯/平塚/茅ケ崎/辻堂/藤沢/大船/戸塚/横浜/川崎/品川/新橋/東京

```
{"station_join":[
   {"station_cd1":1130120,"station_cd2":1130121,
      "station_name1":"湯河原","station_name2":"熱海",
      "lat1":35.146178,"lon1":139.102142,"lat2":35.103573,"lon2":139.077679},
   {"station_cd1":1130119,"station_cd2":1130120,…
```

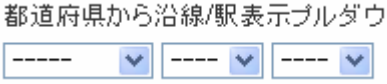
使用例：「都道府県→沿線→駅」のメニューを表示。選択後、選択された内容を alert 表示。

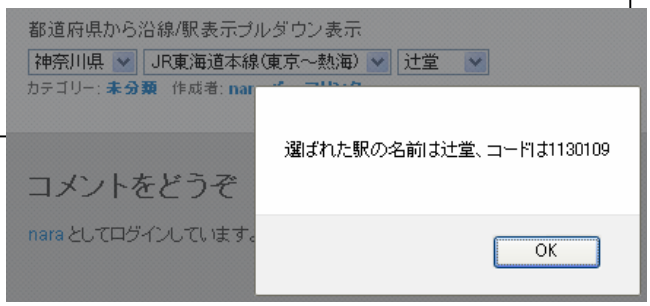
```
都道府県から沿線/駅表示プルダウン表示<br>
<form name="form">
<select name="pref" onChange="setMenuItem(0,this[this.selectedIndex].value)">
   <option value="0" selected>-----
   <option value="1">北海道<option value="2">青森県<option value="3">岩手県
   <option value="4">宮城県<option value="5">秋田県<option value="6">山形県
   <option value="7">福島県<option value="8">茨城県<option value="9">栃木県
   <option value="10">群馬県<option value="11">埼玉県<option value="12">千葉県
   <option value="13">東京都<option value="14">神奈川県<option value="15">新潟県
   <option value="16">富山県<option value="17">石川県<option value="18">福井県
   <option value="19">山梨県<option value="20">長野県<option value="21">岐阜県
   <option value="22">静岡県<option value="23">愛知県<option value="24">三重県
   <option value="25">滋賀県<option value="26">京都府<option value="27">大阪府
   <option value="28">兵庫県<option value="29">奈良県<option value="30">和歌山県
   <option value="31">鳥取県<option value="32">島根県<option value="33">岡山県
   <option value="34">広島県<option value="35">山口県<option value="36">徳島県
   <option value="37">香川県<option value="38">愛媛県<option value="39">高知県
   <option value="40">福岡県<option value="41">佐賀県<option value="42">長崎県
   <option value="43">熊本県<option value="44">大分県<option value="45">宮崎県
   <option value="46">鹿児島県<option value="47">沖縄県
</select>
<select name="s0" onChange="setMenuItem(1,this[this.selectedIndex].value)">
   <option selected>----
</select>
<select name="s1"
   onChange='myFunc(this[this.selectedIndex])'>
   <option selected>----
</select>
</form>
```

```
<script>
function myFunc(station) {
   alert("選ばれた駅の名前は" + station.text + "、コードは" + station.value)
}
var xml = {};

function setMenuItem(type,code){
var s =
   document.getElementsByTagName("head")[0].appendChild(document.createElement("script"));
   s.type = "text/javascript";
   s.charset = "utf-8";

   var optionIndex0 = document.form.s0.options.length; //沿線の OPTION 数取得
   var optionIndex1 = document.form.s1.options.length; //駅の OPTION 数取得

   if (type == 0){
      for ( i=0 ; i <= optionIndex0 ; i++ ){document.form.s0.options[0]=null} //沿線削除
      for ( i=0 ; i <= optionIndex1 ; i++ ){document.form.s1.options[0]=null} //駅削除
      document.form.s1.options[0] = new Option("----",0); //駅 OPTION を空に
      if (code == 0){
         document.form.s0.options[0] = new Option("----",0); //沿線 OPTION を空に
      }else{
         s.src = "http://www.ekidata.jp/api/p/" + code + ".json"; //沿線 JSON データ URL
      }
   }else{
      for ( i=0 ; i <= optionIndex1 ; i++ ){document.form.s1.options[0]=null} //駅削除
      if (code == 0){
        document.form.s1.options[0] = new Option("----",0); //駅 OPTION を空に
      }else{
          s.src = "http://www.ekidata.jp/api/l/" + code + ".json"; //駅 JSON データ URL
      }
   }

   xml.onload = function(data){
      var line = data["line"];
      var station_l = data["station_l"];
      if(line != null){
         document.form.s0.options[0] = new Option("----",0); //OPTION1 番目は Null
         for( i=0; i<line.length; i++){
            ii = i + 1 //OPTION は 2 番目から表示
            var op_line_name = line[i].line_name;
            var op_line_cd = line[i].line_cd;
            document.form.s0.options[ii] = new Option(op_line_name,op_line_cd);
         }
      }
      if(station_l != null){
         document.form.s1.options[0] = new Option("----",0); //OPTION1 番目は Null
         for( i=0; i<station_l.length; i++){
            ii = i + 1 //OPTION は 2 番目から表示
            var op_station_name = station_l[i].station_name;
            var op_station_cd = station_l[i].station_cd;
            document.form.s1.options[ii] = new Option(op_station_name,op_station_cd);
         }
      }
   }
}
</script>
```

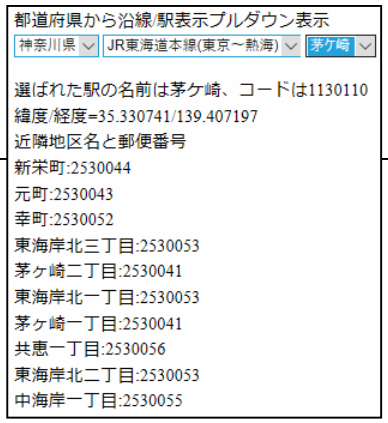
この行だけが重要。
その他の部分はブラックボックスと考えてよい。

都道府県から沿線/駅表示プルダウン表示 →





「都道府県→沿線→駅」のメニューを使ったマッシュアップの例




```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>プログラム</title>
  </head>
  <body>
    都道府県から沿線/駅表示プルダウン表示<br>
    <form name="form">
    <select name="pref" onChange="setMenuItem(0,this[this.selectedIndex].value)">
      <option value="0" selected>-----
      <option value="1">北海道<option value="2">青森県<option value="3">岩手県
      <option value="4">宮城県<option value="5">秋田県<option value="6">山形県
      <option value="7">福島県<option value="8">茨城県<option value="9">栃木県
      <option value="10">群馬県<option value="11">埼玉県<option value="12">千葉県
      <option value="13">東京都<option value="14">神奈川県<option value="15">新潟県
      <option value="16">富山県<option value="17">石川県<option value="18">福井県
      <option value="19">山梨県<option value="20">長野県<option value="21">岐阜県
      <option value="22">静岡県<option value="23">愛知県<option value="24">三重県
      <option value="25">滋賀県<option value="26">京都府<option value="27">大阪府
      <option value="28">兵庫県<option value="29">奈良県<option value="30">和歌山県
      <option value="31">鳥取県<option value="32">島根県<option value="33">岡山県
      <option value="34">広島県<option value="35">山口県<option value="36">徳島県
      <option value="37">香川県<option value="38">愛媛県<option value="39">高知県
      <option value="40">福岡県<option value="41">佐賀県<option value="42">長崎県
      <option value="43">熊本県<option value="44">大分県<option value="45">宮崎県
      <option value="46">鹿児島県<option value="47">沖縄県
    </select>
    <select name="s0" onChange="setMenuItem(1,this[this.selectedIndex].value)">
      <option selected>----
    </select>
    <select name="s1"
      onChange='myFunc(this[this.selectedIndex])'>
      <option selected>----
    </select>
    </form><br>
    <div id=out></div>
    <script>
var xml = {};

function setMenuItem(type,code){
  var s = document.getElementsByTagName("head")[0].appendChild(document.createElement("script"));
  s.type = "text/javascript"; s.charset = "utf-8";
  var optionIndex0 = document.form.s0.options.length; //沿線の OPTION 数取得
  var optionIndex1 = document.form.s1.options.length; //駅の OPTION 数取得
  if (type == 0){
    for ( i=0 ; i <= optionIndex0 ; i++ ){document.form.s0.options[0]=null} //沿線削除
    for ( i=0 ; i <= optionIndex1 ; i++ ){document.form.s1.options[0]=null} //駅削除
    document.form.s1.options[0] = new Option("----",0); //駅 OPTION を空に
    if (code == 0){
      document.form.s0.options[0] = new Option("----",0); //沿線 OPTION を空に
    }else{
      s.src = "http://www.ekidata.jp/api/p/" + code + ".json"; //沿線 JSON データ URL
    }
  }else{
    for ( i=0 ; i <= optionIndex1 ; i++ ){document.form.s1.options[0]=null} //駅削除
    if (code == 0){
     document.form.s1.options[0] = new Option("----",0); //駅 OPTION を空に
    }else{
       s.src = "http://www.ekidata.jp/api/l/" + code + ".json"; //駅 JSON データ URL
  } }
```

次頁に続く

```
    xml.onload = function(data){
      var line = data["line"];
      var station_l = data["station_l"];
      if(line != null){
        document.form.s0.options[0] = new Option("----",0); //OPTION1 番目は Null
        for( i=0; i<line.length; i++){
          ii = i + 1 //OPTION は 2 番目から表示
          var op_line_name = line[i].line_name;
          var op_line_cd = line[i].line_cd;
          document.form.s0.options[ii] = new Option(op_line_name,op_line_cd);
      } }
      if(station_l != null){
        document.form.s1.options[0] = new Option("----",0); //OPTION1 番目は Null
        for( i=0; i<station_l.length; i++){
          ii = i + 1 //OPTION は 2 番目から表示
          var op_station_name = station_l[i].station_name;
          var op_station_cd = station_l[i].station_cd;
          document.form.s1.options[ii] = new Option(op_station_name,op_station_cd);
} } } }
// ---------------------------------------------
var outp = document.getElementById("out");

function myFunc(station) {
    outp.innerHTML = "選ばれた駅の名前は" + station.text + "、コードは" + station.value + "<br>";
    getPosition(station.value);
}

// ----------------  マッシュアップ(1) ekidata  駅コード->  緯度/経度  ------------------
function getPosition(scode) {
    xml = {};
    var script = document.createElement('script');
    script.src = "http://www.ekidata.jp/api/s/" + scode + ".json";
    document.body.appendChild(script);
    xml.onload = dply; }

function dply(data) {
    outp.innerHTML += "緯度/経度="+data.station[0].lat+"/"+data.station[0].lon + "<br>"
    findZip(data.station[0].lat,data.station[0].lon);
}
// ------  マッシュアップ(2) heartrails geoapi  緯度/経度  ->  近隣地区と郵便番号  --------
function findZip(lat,lon) {
    var target = document.createElement('script');
    target.charset = 'utf-8';
    target.src = "http://geoapi.heartrails.com/api/json"
                    + "?method=searchByGeoLocation"
                    + "&y=" + lat
                    + "&x=" + lon
                    + "&jsonp=callbackf";
    document.body.appendChild(target);
}

function callbackf(data) {
    var o = "近隣地区名と郵便番号<br>";
    for (var i in data.response.location)
        o += data.response.location[i].town+":"+data.response.location[i].postal+"<br>";
    outp.innerHTML += o;
}
    </script>
  </body>
</html>
```