D3.jsを使った
データビジュアライズ勉強会

2016年6月18日

先端IT活用推進コンソーシアム
クラウド・テクノロジー活用部会

タイムスケジュール

- 14:00〜15:00
  - D3.js入門
- 15:00〜16:45
  - オープンデータをD3.jsで可視化
- 16:45〜17:45
  - 自分で探したデータを可視化
- 17:45〜18:00
  - 可視化したものを見てみよう

当初の予定より、ゆっくり進めます

本日の目標

- D3.jsを使ったデータの可視化を理解する
  - 数値を眺めているだけでは見えなかったものが、可視化によって何かが見えてきたらステキ
- オープンデータを可視化してみる
  - 自分の住んでいる場所や実家のデータを探す
  - そのデータを可視化してみる

- 注意事項
  - 今日は、WindowsもMacも、Firefoxを使うこと

実は

- 別の方法でもできるけど
  - Google Maps
    - https://support.google.com/mymaps/answer/3024836?hl=ja
  - Microsoft Excel の Power View
    - http://office.microsoft.com/ja-jp/excel-help/HA102899553.aspx
- D3.jsを使うメリット
  - 思い通りの表現・拡張ができる
    - アニメーションによって時間による変化も可視化
  - 地図以外にも活用できる
  - オンラインとオフラインを明確に区別できるので
    - 業務データでも実施する事ができる

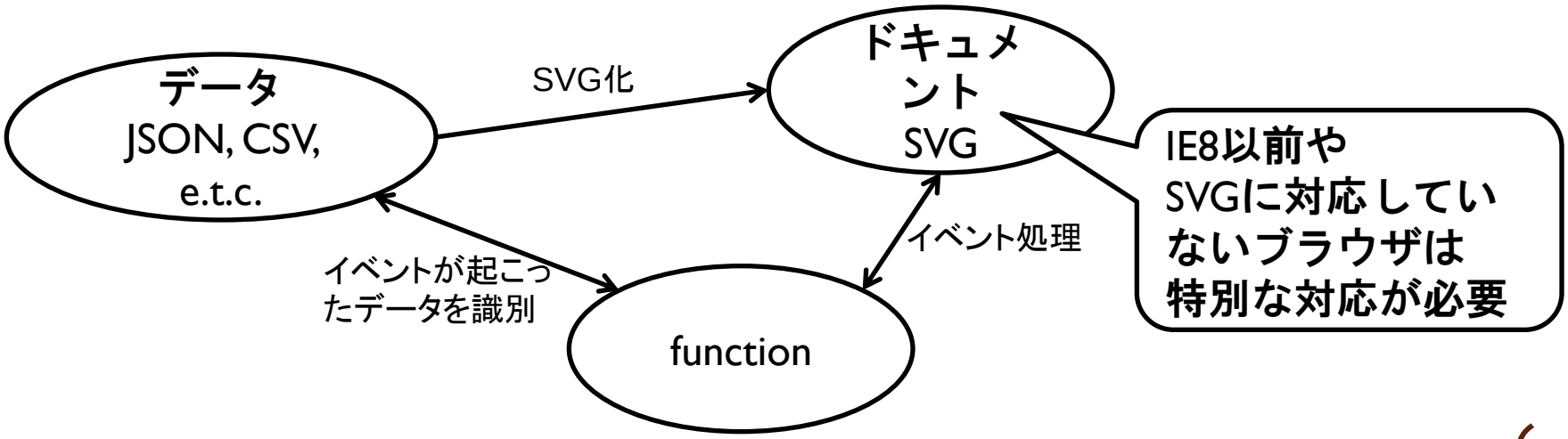
D3.js入門

D3.js とは

D3.js（またはD3:Data-Driven Documents、旧:Protovis[1]）は、2011年に開発が始まった[2]ウェブブラウザで動的コンテンツを描画するJavaScriptライブラリである。World Wide Web Consortium準拠のデータ可視化ツールとして、Scalable Vector Graphics（SVG）、JavaScript、HTML5、Cascading Style Sheetsを最大限に活用している。その他多くのライブラリとは対照的に、最終的に出力された結果に視覚的な調整ができる。[3]

ウィキペデアより

- **データをドキュメント化して、関係も保持**
  - **SVGを効率良く生成するためのライブラリです**



D3.jsを使ってビジュアライズ

- 今回は地図を軸に可視化してみる
  - D3.js は、地図描画の機能が充実

- 注意：地図は楽だけど、D3.jsでグラフを描くのは面倒
  - 必要な部品がそろっているだけ
  - D3.js Examples → 各グラフを描くためのソースを参照
    - https://github.com/mbostock/d3/wiki/Gallery
  - 参考URL http://postd.cc/what-d3js-is-not/
  - グラフを書くのなら、グラフ用ライブラリを使用した方が良い
    - →次回のハンズオンでやる予定
- 参考書籍

  エンジニアのための データ可視化[実践]入門
  ~D3.jsによるWebの可視化
  - http://www.amazon.co.jp/dp/4774163260
  - D3.js はほとんど出てこないけど、「可視化」の理解が深まります

開発環境の準備

- D3.js をダウンロード
  - http://d3js.org/
    - d3.zip をダウンロードし、d3.min.js を取り出す
    - 今回は、すでにd3ディレクトリに入っているので不要
- テキストエディタ
  - お気に入りのテキストエディタでOK
- ブラウザ
  - [F-12]を押せば、開発ツールが起動する
  - **Firefox**＋firebug：「ツール」→「Web開発」→「Firebug」→「Firebugを開く」
  - **Safari**：「環境設定」→「詳細」→「メニューバーに"開発"メニューを表示」
  - IE（9以降）：「F12 開発者ツール」
    - ローカルだとうまく動作しない
      - データにアクセスしている部分をjQueryにすれば、動くらしい
  - Chrome：「ツール」→「デベロッパーツール」
    - ローカルのデータファイルを参照するため、起動オプションを追加
      「--allow-file-access-from-files」

SVG(Scalable Vector Graphics)について

- Wikipedia
  - http://ja.wikipedia.org/wiki/Scalable_Vector_Graphics

> **Scalable Vector Graphics**（スケーラブル・ベクター・グラフィックス、**SVG**）は、XMLをベースとした、2次元ベクターイメージ用の画像形式の1つである。アニメーションやユーザインタラクションもサポートしている。SVGの仕様はW3Cによって開発され、オープン標準として勧告されている。

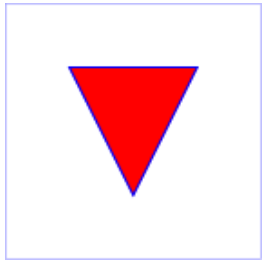
- SVG仕様
  - https://triple-underscore.github.io/SVG11/

- 使用上の注意
  - 順番通りに上に重ねて描画
  - 対応していないブラウザだと、何も表示されない
  - 注意：ブラウザによっては、微妙に見え方が違うかも

SVG(Scalable Vector Graphics)について

- **代表的な図形：sample_svg.html参照**

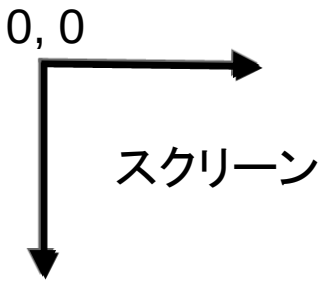
```
<rect  x="200"  y="50"  width="400"  height="200"
    fill="yellow"  stroke="navy"  stroke-width="10"  />

<circle  cx="600"  cy="200"  r="100"  fill="red"  stroke="blue"
  stroke-width="10"  />

<path  d="M 100 100 L 300 100 L 200 300 z"  fill="red"
    stroke="blue"  stroke-width="3" />

<line  x1="100"  y1="300"  x2="300"  y2="100" stroke="green"
  stroke-width="5" />

<text  x="250"  y="150"  font-family="Verdana"  font-size="55"  fill="blue"
  >  Hello, out there  </text>
```



  - **注意：後ろに書いたものが上にくる**
- **D3.jsは、データからSVG（HTML）を自動生成する**

SVGで絵を描く上での注意

- 座標は、左上がx=0, y=0
  - 右下に向かって、数値が増えていく



- 色は、名前か”#000000” で指定
  - http://www.hi-ho.ne.jp/douton/htmlcolor.html
    - 赤：red　　#ff0000
    - 緑：green　　#008000
    - 青：blue　　#0000ff
    - 黒：black　　#000000
    - 白：white　　#ffffff
    - 灰：gray　　#808080

step1.html：SVGサンプル

```
<!DOCTYPE html>                    ← HTML5で記述することを宣言
<html lang="ja">                   ← html部
<head>                             ← ヘッダ部
<meta charset="SJIS">   ← このファイルの文字コードはSJIS
<title>Sample SVG</title>
</head>
<body>                             ← ボディ部
<svg width="1000" height="1000">          ← SVG領域を1000x1000で確保
  <rect   x="200"  y="50"   width="400" height="200" fill="yellow"  stroke="navy" stroke-width="10" />
                          ← SVG 領域内に矩形を描く
  <circle  cx="600"  cy="200"  r="100"    fill="red"  stroke="blue"
     stroke-width="10" />                          ← SVG領域内に円を描く
  <line   x1="100" y1="300" x2="300"   y2="100"    stroke="green" stroke-width="5" />
  <path   d="M 100 100 L 300 100 L 200 300 z" fill="red" stroke="blue" stroke-width="3" />
  <text   x="250"  y="150"  font-family="Verdana"  font-size="55"  fill="blue" >
Hello, SVG World</text>
</svg>
</body>
</html>
```

課題－１

- step1.html を修正
  - 文字を箱の中に収まるようにする
  - 文字を自分の名前に修正
    - 保存する文字コードに注意
  - 車の絵にする

課題－１、回答例

- http://free-illustrations.gatag.net/2015/03/08/160000.html

←SVG形式で
ダウンロード可能

D3.jsの超概要 http://ja.d3js.node.ws/

- セレクタ（W3C Selectorsを参照）
  - d3.select(**“#hoge”**) → <xxx id=“hoge”> を対象
  - d3.select(**“.hoge”**) → <xxx class=“hoge”>を対象
  - d3.select(**“hoge”**) → <hoge>を対象

```
signal = [
    { “cx”: 100, “cy”: 100, “color”: “#0000ff”, “title”:”青”,},
    { “cx”: 200, “cy”: 100, “color”: “#ffff00”, “title”:”黄”,},
    { “cx”: 300, “cy”: 100, “color”: “#ff0000”, “title”:”赤”,},
];
```

- セレクション
  - selectAll(), enter(), exit()
  - 繰り返し処理が楽に書ける
    - d3.select("#TEXT1").**selectAll("p")**.style("color", “#000000");
- 動的プロパティ
    - svg.selectAll(".node").data(signal).text(**function(d) { return d.title; }**);
    - svg.selectAll(".node").data(signal).text(**function(d, i) { return i; }**);
- データの結合
  - 追加：svg.selectAll(".node").data(signal).**enter()**.append(“text").text(“piyopiyo”);
  - 更新：svg.selectAll(".node").data(signal).text(“hogehoge”);
  - 削除：svg.selectAll(".node").data(signal).**exit()**.remove();

その他の今回使用するメソッド

- アニメーション
  - transaction()
- 外部リソースの読み込み
  - d3.csv(), d3.json(), d3.text()
- 地図描画
  - d3.geo.mercator()
  - d3.geo.path().projection(projection);

step2.html：D3.js入門

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="SJIS">
<script src="./d3/d3.min.js"></script>   ← d3.jsの読み込みを指定
<script src="./step2.js"></script>       ← step2.jsの読み込みを指定
</head>
<body>
<div id="view"></div>                    ← 描画領域を定義
</body>
</html>
```

以降は、jsファイルの指定が違うだけ

step2.js：D3.js入門、前半

```
var width = 900, height = 650;
var svg = null;
var data = null;

window.onload = function() {

    svg = d3.select("#view").append("svg")  // <div id="view">に<svg>を追加
            .attr("width", width)
            .attr("height", height);

    data = [
            { "cx": 100, "cy": 100, "color": "#0000ff",},      // 青
            { "cx": 200, "cy": 100, "color": "#ffff00",},      // 黄
            { "cx": 300, "cy": 100, "color": "#ff0000",},      // 赤
    ];
```

step2.js：D3.js入門、後半

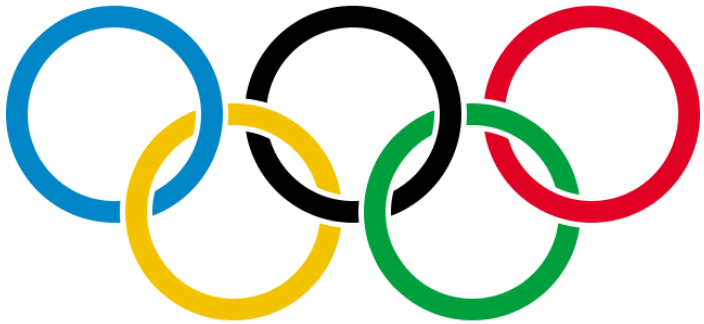
```
    svg.selectAll(".node")      //  <circle class="node">を選択
            .data(data).enter()        // データの増分を対象
            .append("circle")          // svgのcircleを追加
            .attr("class", "node")
            .attr("cx", function(d) { return d.cx; })          // 円の中心
            .attr("cy", function(d) { return d.cy; })          // 円の中心
            .attr("r", 30)                                     // 円の半径
            .style("stroke", function(d) { return "#000000"; }) // 枠線の色
            .style("stroke-width", 1)                          // 枠線の太さ
            .style("fill", function(d) { return d.color; })    // 塗りつぶしの色
            .on('click', function(d) {                                  // クリック時の処理
                    console.log(d);
            })
            ;
}
```

実行後のHTMLの見方と、console.log の解説

課題－２

- step2.js を修正
  - オリンピックマークを描いてみる
    - 微妙な重なりまでは表現する必要なし
    - １．丸を５つにする
    - ２．位置を修正する
    - ３．色を指定する
    - ４．縁を太くする
    - ５．塗りつぶしから、縁色に変更する
    - ６．塗りつぶしを透明にする
      - jsに追加

        .style("fill-opacity",0)　　// 透明
    - 書き方が分からないことは、SVGの仕様を参照



正解は、step3.js にあります

アニメーション

- ある状態からある状態に値を連続的に変化
  - 位置
  - 大きさ
  - 色
    - HTMLではrgb(255,255,255)で表記
- jsの書き方

```
d3.selectAll(“.node”)          // 対象を指定
.transition()                  // アニメーション
.delay(500)                    // 開始までの遅延時間
.duration(1000)          // 変化にかかる時間
.attr(“cx”, 200)         // 変化後の値
.attr(“cy”, 125)         // 変化後の値
```

step3.js：アニメーション、前半

```
var width = 900, height = 650;
var svg = null;
var data = null;

window.onload = function() {

    svg = d3.select("#view").append("svg")  // <div id="view">に<svg>を追加
            .attr("width", width)
            .attr("height", height);

    // オリンピックマークを描いて、アニメーションする
    data = [
            { "cx": 100, "cy": 100, "color": "#0000ff",},          // 青 data[0]
            { "cx": 200, "cy": 100, "color": "#000000",},          // 黒 data[1]
            { "cx": 300, "cy": 100, "color": "#ff0000",},          // 赤 data[2]
            { "cx": 150, "cy": 150, "color": "#ffff00",},          // 黄 data[3]
            { "cx": 250, "cy": 150, "color": "#00ff00",},          // 緑 data[4]
    ];
```

step3.js：アニメーション、後半

```
    svg.selectAll(".node")      //  <circle class="node">を選択
            .data(data).enter()       // データの増分を対象
            .append("circle")         // svgのcircleを追加
            .attr("class", "node")
            .attr("cx", function(d) { return d.cx; })          // 円の中心
            .attr("cy", function(d) { return d.cy; })          // 円の中心
            .attr("r", 45)                                     // 円の半径
            .style("stroke", function(d) { return d.color; })  // 枠線の色
            .style("stroke-width", 10)                         // 枠線の太さ
            .style("fill", function(d) { return "#ffffff"; })  // 塗りつぶしの色
            ;
    svg.selectAll(".node")      //  <circle class="node">を選択
            .transition()                                      // アニメーション
            .delay(500)                                        // 0.5秒後に開始
            .duration(1000)                                    // 1秒間で変化完了
            .attr("cx", 200)                                   // 円の中心を変更
            .attr("cy", 125)
            .transition()                                      // アニメーション
            .duration(1000)                                    // 1秒間で変化完了
            .attr("cx", function(d) { return d.cx; })          // 円の中心を元に戻す
            .attr("cy", function(d) { return d.cy; })
            ;
}
```

課題－３

- もっと格好良いアニメーションにする
  - ３つの丸から５つの丸に変化
    - 開始時は、２つの丸の位置を同じにする
    - アニメーション前に、一部のデータを書き換える
      - data[3].cx = 150; data[3].cy = 150;
      - data[4].cx = 250; data[4].cy = 150;
    - 余力のある人は、r や stroke-width も変化させる

外部リソースを参照

- データを外部から参照する
  - csv, json, xml → d3.csv(), d3.json(), d3.text()
- APIも利用できる
  - リアルタイムデータをD3.jsで表示できる
    - **注意：許可されているAPIのみ直接参照が可能**
      - HTTPヘッダに「Access-Control-Allow-Origin: *」があるか？
  - 欲しい形式でデータが取れない場合
    - JavaScriptで処理する
    - 外部ツールで処理するのもアリ
      - Excel, テキストエディタ, UNIXコマンド

MacのSafariでローカルファイルにアクセス
「開発」→「ローカルファイルの制限を無効にする」

step4.js：外部リソース参照

```
var csvfile = "./step4.csv";      // ファイル名を指定
                                  // http://.... という指定も可能
d3.csv(csvfile, function(data) {  // CSVファイルの中身が、変数dataに入ってくる

        svg.selectAll(".node")      //  <circle class="node">を選択
                .data(data).enter()       // データの増分を対象
                .append("circle")         // svgのcircleを追加
                .attr("class", "node")
                .attr("cx", function(d) { return d.cx; })          // 円の中心
                .attr("cy", function(d) { return d.cy; })          // 円の中心
                .attr("r", 45)                                     // 円の半径
                .style("stroke", function(d) { return d.color; })  // 枠線の色
                .style("stroke-width", 10)                         // 枠線の太さ
                .style("fill", function(d) { return "#ffffff"; })  // 塗りつぶしの色
                .style("fill-opacity", 0)                          // 透明
                ;
    });
}
```

step4.csv：外部リソース

```
cx,cy,color                ← １行目はヘッダ
100,100,“#0000ff“          ← ２行目以降はデータ
200,100,"#000000"
300,100,"#ff0000"
150,150,"#ffff00"
250,150,"#00ff00"
```

課題－４

- 機能追加
  - 半径(r)、線の太さ(stroke-width)を指定
- データ修正
  - 別の絵のデータを作ってみてください

ここまで出来たら

- 天才ハンス・ロスリングと同じ事ができる？！
  - http://matome.naver.jp/odai/2137432558736685501
  - http://www.gapminder.org/world/
  - ↑これはFlashだけど、同じ事は可能
- アニメーションや特殊効果も作成し放題
  - 絵が下手な人でも、試行錯誤で作れる
  - D3.jsを使ったサンプル
    - http://aramoto.sakura.ne.jp/IoT/graph3.php
    - http://aramoto.sakura.ne.jp/sparql/

まずは、地図を準備しよう

地図データの準備

- D3.js の geo パッケージで読める形式が必要
  - GeoJSON：地理情報用に規格されたJSONデータ形式
  - TopoJSON：GeoJSONの拡張形式。D3.jsではプラグインが必要
    - 冗長性を排するので、データサイズが20%程度になる
- Shape形式の地図を入手
  - 入手元：国土交通省、国土数値情報
    - http://nlftp.mlit.go.jp/ksj/gml/datalist/KsjTmplt-N03.html
    - 神奈川県：N03-130401_14_GML.zip
- GeoJSON形式に変換
  - このサイトで10%に圧縮＋GeoJSONに変換
    - http://www.mapshaper.org/
    - 変換手順
      - １．「select」ボタンを押して、shpファイルをアップロード
      - ２．「simplify」→「Apply」を押して、圧縮率を決める
      - ３．「Export」を押して、「GeoJSON」を指定
    - 10%というのはShapefileでの比（1,191KB→131KB）
    - Shapefile→GeoJSONに変換すると大きくなる（131KB→514KB）
- 変換結果（GeoJSON形式、514KB）
  - http://cloud.aitc.jp/20140717_D3js/gis/kanagawa.json

緯度・経度を扱う上での注意

0, 0

スクリーン

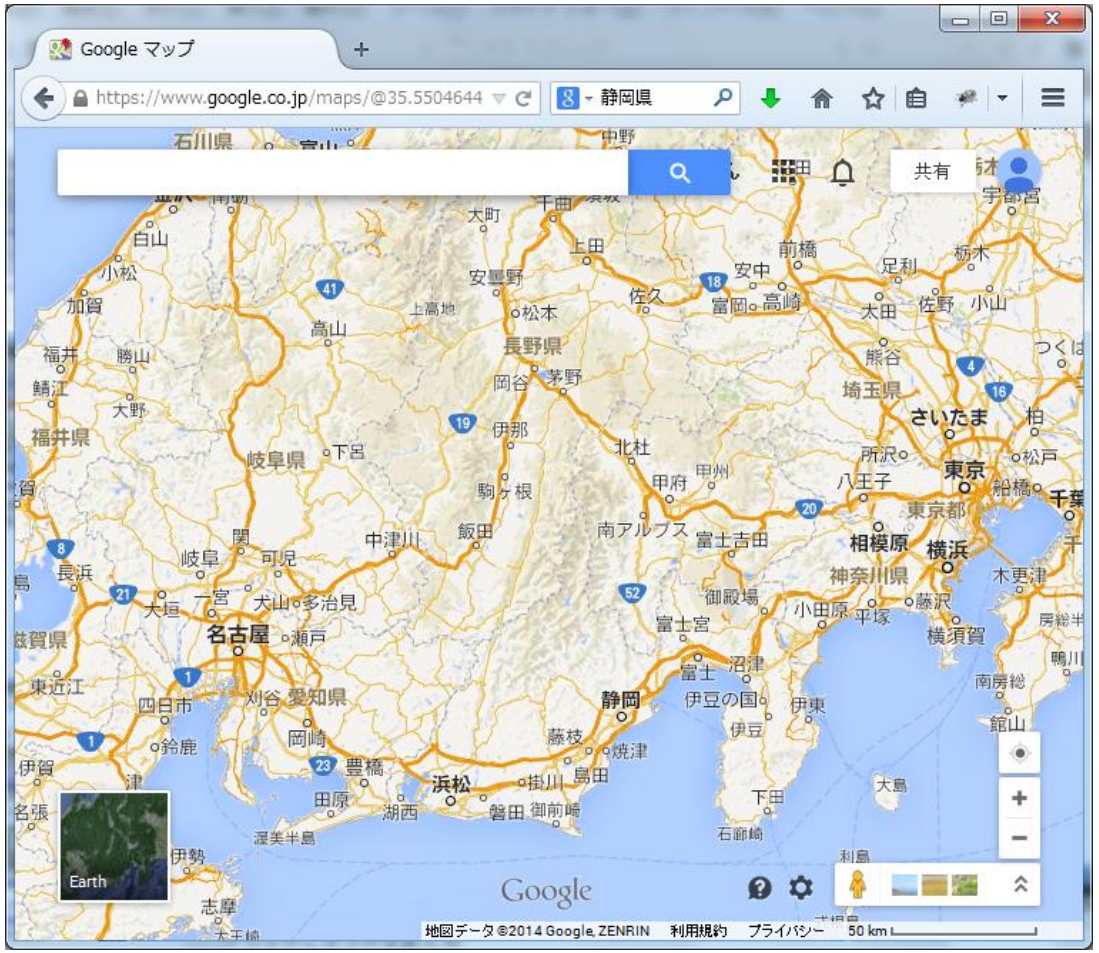
画面内に描画するため
・スクリーンの原点と、緯度・経度の原点が違う
・表示スケールの調整

経度・緯度

136.09, 34.36

0 , 0
136.09, 36.77

900 , 0
140.18, 36.77



0 , 600
136.09, 34.36

900 , 600
140.18, 34.36

step5.js：D3.jsで地図を描画

```
map = svg.append("g");                          // <svg>に<g>を追加

// 地図データを読み込む
d3.json("./gis/kanagawa.json",function(json) {
        // 緯度・経度からスクリーン座標に変換
        var projection = d3.geo.mercator()
                        .scale(30000)                   // 縮尺を指定
                        .center(d3.geo.centroid(json))  // 中心位置を計算
                        .translate([width / 2, height / 2]);    // 地図の中心を画面の中心にする

        // projectionを使ってpathを変換
        var path = d3.geo.path().projection(projection);

        // 地図を描画
        map.selectAll("path")
        .data(json.geometries).enter()
        .append("path")
        .attr("d", path)
        .attr("fill","#C0C0C0")
        .attr("stroke","#000000")
        ;
});
```

課題－５

- 自分の出身地の地図を表示
- 東京の地図を表示してみる
  - 何故、今回は東京を題材にしなかったのか？
- 余裕のある人は、海外にもトライ

オープンデータから
表示用ファイルを準備

横浜市のオープンデータ

- 横浜オープンデータポータル
  - http://data.yokohamaopendata.jp/
- ヨコハマ・アート・LOD
  - http://fp.yafjp.org/yokohama_art_lod
- よこはまオープンデータカタログ（試行版）
  - http://www.city.yokohama.lg.jp/seisaku/seisaku/opendata/catalog.html
    - 2014年7月14日オープン

SPARQLによるクエリが可能

オープンデータじゃないけど

- 横浜市統計情報ポータルサイト
  - http://www.city.yokohama.lg.jp/ex/stat/

その他のオープンデータ

- Open DATA METI
  - http://datameti.go.jp/data/ja/dataset
- データカタログサイト（試行版）
  - http://www.data.go.jp/
- オープンデータプラットフォーム
  - http://odp.jig.jp/
- 気象庁XML公開サイト
  - http://api.aitc.jp/
- 探し方
  - Google「東京都 オープンデータ」で検索

SPARQLによるクエリが可能

SPARQLによるクエリ例

- **オープンデータプラットフォーム** http://odp.jig.jp/
  - **SPARQLで横浜市に関する登録を検索**
    - http://sparql.odp.jig.jp/sparql.html

```
select * where {
  ?id ?t <http://odp.jig.jp/odp/1.0#OpenDataCity> .
  ?id ?p ?o .
  filter(regex(?o, "横浜市"))
} limit 10
```

    - **神奈川県横浜市：**http://odp.jig.jp/jp/jig/opendatacity/13
    - **神奈川県横浜市金沢区：**http://odp.jig.jp/jp/jig/opendatacity/5

敷居はかなり高いけど、
慣れたら「RDBを公開してくれている」のと同じ感覚

データを準備する上での注意－１

- 文字コード
  - CSVはShift-JISで公開されているものが多い
  - JavaScriptで読み込む場合
    - とりあえず「UTF-8」に変換すれば、トラブルは起こりにくい

- ライセンス
  - データを公開しているサイトの利用規約を参照
    - http://data.yokohamaopendata.jp/dataset/kanazawa_kz-opendata_2013_35-child-care-event

    →リソース→ライセンスを参照

データを準備する上での注意－2

- 他サイト上のデータを直接参照したい
  - 常に最新データを表示できる
    - 今日やるようなデータ加工をJavaScriptで実装
  - CORS（Cross-Origin Resource Sharing）
    - データの公開サイトが、他サイトから直接参照を許可しているか？
    - 確認方法
      - HTTPヘッダに「Access-Control-Allow-Origin: *」があればOK
- 開発時に、全部ローカルでやる場合は
  - Firefox, Safariは、特に何もしなくてもOK
  - IEはうまく動作しない
    - ファイルアクセス部分にjQueryを使う事で回避可能
  - Chromeは起動オプションを追加
    - 「--allow-file-access-from-files」

使用するデータを選択

- 使用するデータを選択する時、注意する点
  - 住所もしくは緯度経度と、数値があるもの
  - 時間とともに変化すると、さらに面白い
- 今回使用するデータ
  - 横浜市総務局 防災関連データ
    - http://www.city.yokohama.lg.jp/somu/org/kikikanri/data/

    →「地域防災拠点」 **※注：XMLファイル**

    クリエイティブコモンズ「表示」(CC BY)
      - 二次著作物を作成する場合は、利用するデータの出典（データのタイトルと著作権者名など）の表示をお願いします。
- 使用するツール
  - Excel で編集を行います
  - ExcelとJavaScriptに自信があれば、他データに挑戦してもOK
  - Excelを持っていない人は
    - Sample ディレクトリの下からコピーしてください

データ準備－１

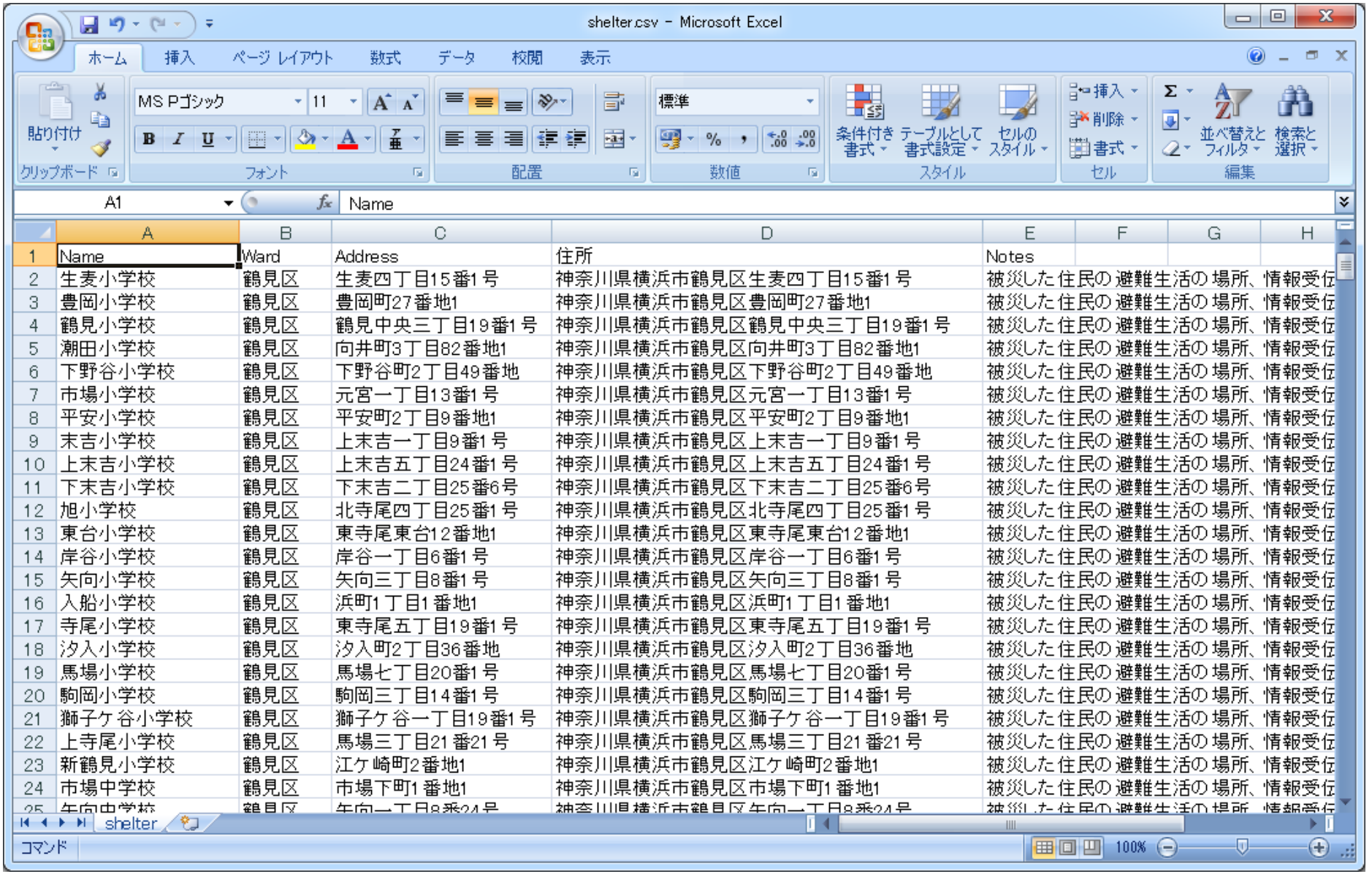
- XML→CSVに変換：Excelを利用する方法
  - Excel でダウンロードしたファイルを開く
    - 「XMLテーブルとして開く」を選択
      - １回目はちょっと時間がかかります
  - 「名前を付けて保存」でCSVを選択
    - ファイル名は「shelter.csv」
- XML→CSVに変換：Webを利用する方法
  - 以下のサイトを利用する
    - http://www.luxonsoftware.com/converter/xmltocsv
    - XMLをアップロードし、「Convert XML to CSV」を押す
    - 「Download Result」を押す
    - ダウンロードしたzipファイルを解凍する
    - テキストエディタで、文字コードをUTF-8→Shift_JISに変換する

データ準備－２

- CSVを加工
  - Excelを再起動し、「shelter.csv」を開きなおす
  - フルの住所のカラムをCとDの間に追加
    - タイトルは「住所」
    - =“神奈川県横浜市” & B2 & C2
  - 他のデータを使う場合も
    - ゴミを削除して、CSVとして正しい形式にする

- Excelを持っていない人は
  - sample ディレクトリの下から「 shelter_住所あり.csv」をコピー

データ準備－２－最終形



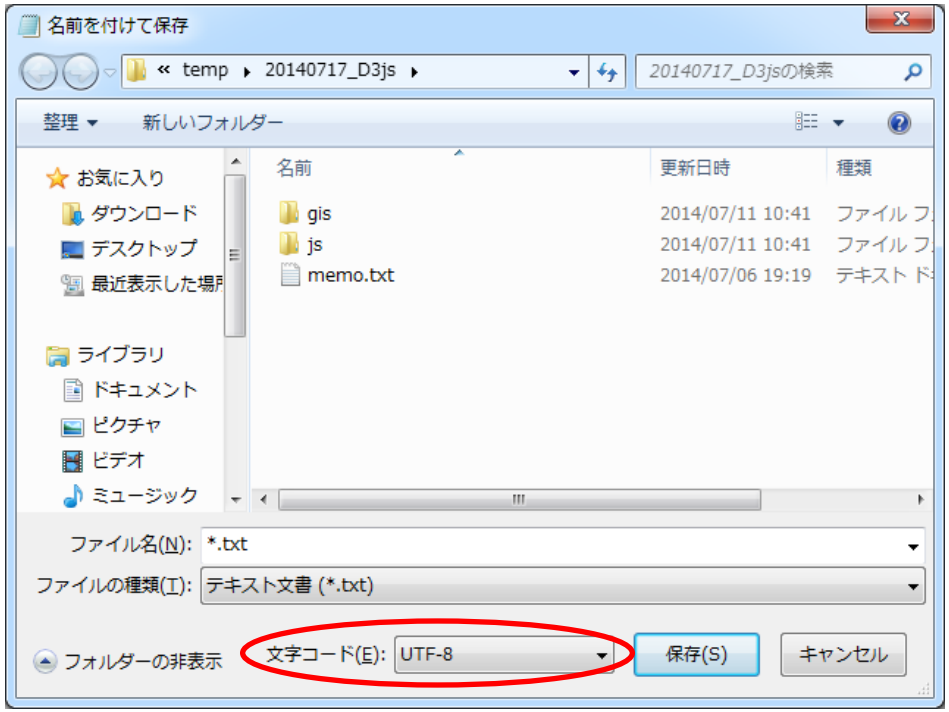
	A	B	C	D	E	F	G	H
1	Name	Ward	Address	住所	Notes			
2	生麦小学校	鶴見区	生麦四丁目15番1号	神奈川県横浜市鶴見区生麦四丁目15番1号	被災した住民の避難生活の場所、情報受伝			
3	豊岡小学校	鶴見区	豊岡町27番地1	神奈川県横浜市鶴見区豊岡町27番地1	被災した住民の避難生活の場所、情報受伝			
4	鶴見小学校	鶴見区	鶴見中央三丁目19番1号	神奈川県横浜市鶴見区鶴見中央三丁目19番1号	被災した住民の避難生活の場所、情報受伝			
5	潮田小学校	鶴見区	向井町3丁目82番地1	神奈川県横浜市鶴見区向井町3丁目82番地1	被災した住民の避難生活の場所、情報受伝			
6	下野谷小学校	鶴見区	下野谷町2丁目49番地	神奈川県横浜市鶴見区下野谷町2丁目49番地	被災した住民の避難生活の場所、情報受伝			
7	市場小学校	鶴見区	元宮一丁目13番1号	神奈川県横浜市鶴見区元宮一丁目13番1号	被災した住民の避難生活の場所、情報受伝			
8	平安小学校	鶴見区	平安町2丁目9番地1	神奈川県横浜市鶴見区平安町2丁目9番地1	被災した住民の避難生活の場所、情報受伝			
9	末吉小学校	鶴見区	上末吉一丁目9番1号	神奈川県横浜市鶴見区上末吉一丁目9番1号	被災した住民の避難生活の場所、情報受伝			
10	上末吉小学校	鶴見区	上末吉五丁目24番1号	神奈川県横浜市鶴見区上末吉五丁目24番1号	被災した住民の避難生活の場所、情報受伝			
11	下末吉小学校	鶴見区	下末吉二丁目25番6号	神奈川県横浜市鶴見区下末吉二丁目25番6号	被災した住民の避難生活の場所、情報受伝			
12	旭小学校	鶴見区	北寺尾四丁目25番1号	神奈川県横浜市鶴見区北寺尾四丁目25番1号	被災した住民の避難生活の場所、情報受伝			
13	東台小学校	鶴見区	東寺尾東台12番地1	神奈川県横浜市鶴見区東寺尾東台12番地1	被災した住民の避難生活の場所、情報受伝			
14	岸谷小学校	鶴見区	岸谷一丁目6番1号	神奈川県横浜市鶴見区岸谷一丁目6番1号	被災した住民の避難生活の場所、情報受伝			
15	矢向小学校	鶴見区	矢向三丁目8番1号	神奈川県横浜市鶴見区矢向三丁目8番1号	被災した住民の避難生活の場所、情報受伝			
16	入船小学校	鶴見区	浜町1丁目1番地1	神奈川県横浜市鶴見区浜町1丁目1番地1	被災した住民の避難生活の場所、情報受伝			
17	寺尾小学校	鶴見区	東寺尾五丁目19番1号	神奈川県横浜市鶴見区東寺尾五丁目19番1号	被災した住民の避難生活の場所、情報受伝			
18	汐入小学校	鶴見区	汐入町2丁目36番地	神奈川県横浜市鶴見区汐入町2丁目36番地	被災した住民の避難生活の場所、情報受伝			
19	馬場小学校	鶴見区	馬場七丁目20番1号	神奈川県横浜市鶴見区馬場七丁目20番1号	被災した住民の避難生活の場所、情報受伝			
20	駒岡小学校	鶴見区	駒岡三丁目14番1号	神奈川県横浜市鶴見区駒岡三丁目14番1号	被災した住民の避難生活の場所、情報受伝			
21	獅子ケ谷小学校	鶴見区	獅子ケ谷一丁目19番1号	神奈川県横浜市鶴見区獅子ケ谷一丁目19番1号	被災した住民の避難生活の場所、情報受伝			
22	上寺尾小学校	鶴見区	馬場三丁目21番21号	神奈川県横浜市鶴見区馬場三丁目21番21号	被災した住民の避難生活の場所、情報受伝			
23	新鶴見小学校	鶴見区	江ケ崎町2番地1	神奈川県横浜市鶴見区江ケ崎町2番地1	被災した住民の避難生活の場所、情報受伝			
24	市場中学校	鶴見区	市場下町1番地1	神奈川県横浜市鶴見区市場下町1番地1	被災した住民の避難生活の場所、情報受伝			
25	矢向中学校	鶴見区	矢向一丁目8番24号	神奈川県横浜市鶴見区矢向一丁目8番24号	被災した住民の避難生活の場所、情報受伝			

データ準備－３

- 住所を緯度経度に変換する
  - 地図に描画するためには、緯度・経度が必要
- 外部のWebで変換可能
  - 外部に漏れても良いデータのみ
  - http://newspat.csis.u-tokyo.ac.jp/geocode/
  - 「今すぐサービスを利用する」をクリック
  - 各パラメータを設定
    - 住所を含むカラム番号：4
    - 変換したいファイル名： shelter.csv
  - 「送信」で、変換結果のCSVが落ちてくる
    - ダウンロード後、Excelで開いて内容を確認
    - 取得した緯度経度をGoogleMapsで確認
      - 「fY fX」の順で検索

データ準備－４

- 文字コード変換
  - テキストエディタで開き、文字コードを「UTF-8」に変更し保存
  - ファイル名は、「shelter_utf8.csv」を指定



Windowsのメモ帳
「ファイル」→「名前を付けて保存」

Macのテキストエディット
「テキストエディット」→「環境設定」

データ準備－５（余裕のある人だけ）

- 「地域防災拠点」以外のデータに対して、同様の手順を行う
  - 「住所を含むカラム番号」に注意

- 改行が正しく表示されない場合
  - 住所→緯度経度変換時にUNIX改行に変わった
  - 心配なら、別のエディタでCR+LFに変換
    - 変換しなくても、特に問題はない

step6.js：緯度・経度で描画

```
    // 金沢区内公園を表示
//  var csvfile = "http://cloud.aitc.jp/20140717_D3js/23-kz-park_utf8.csv";
    var csvfile = "./23-kz-park_utf8.csv";
    d3.csv(csvfile,function(csv) {
        svg = d3.select("#view").append("svg")  // <div id="view">に<svg>を追加
                .attr("width",width)
                .attr("height",height)
                .attr("viewBox","0 0 2000 2000") // 座標データの最小値と幅（最大値ー最小値）
                ;
        svg.selectAll(".node")      //  <circle class="node">を選択
                .data(csv).enter()       // データの増分を対象
                .append("circle")          // svgのcircleを追加
                .attr("class","node")
                .attr("cx",function(d) { return (d.経度*10000 - 1395000);})      // 円の中心
                .attr("cy",function(d) { return (354000 - d.緯度*10000);})       // 円の中心
                .attr("r",10)                                                    // 円の半径
                .style("stroke",function(d) { return "#000000";})                // 枠線の色
                .style("stroke-width","1px")                                     // 枠線の太さ
                .style("fill","white“)                                           // 塗りつぶしの色
                ;
    });
}
```

課題－６

- 作成したデータ「地域防災拠点」を描画
  - ファイル名を修正
  - d.緯度 → d.fY
  - d.経度 → d.fX

step7.js：地図上にデータを描画、前半

```
var map = svg.append("g");                    // <svg>に<g>を追加

// 地図データを読み込む
//  d3.json("http://cloud.aitc.jp/20140717_D3js/gis/kanagawa.json",function(json) {
    d3.json("./gis/kanagawa.json",function(json) {

                    // 緯度・経度からスクリーン座標に変換
                    var projection = d3.geo.mercator()
                                .scale(50000)                    // 縮尺を指定
                                .center(d3.geo.centroid(json))   // 中心位置を計算
                                .translate([width / 2, height / 2]);   // 地図の中心を画面の中心にする

                    // projectionを使ってpathを変換
                    var path = d3.geo.path().projection(projection);

                    // 地図を描画
                    map.selectAll('path')
                    .data(json.geometries)
                    .enter()
                    .append('path')
                    .attr('d', path)
                    .attr("fill","#C0C0C0")
                    .attr("stroke","#000000" )
                    .on('click',function(d) {
                                console.log(d);
                    });
```

step7.js：地図上にデータを描画、後半

```
// 金沢区内公園 http://www.city.yokohama.lg.jp/kanazawa/kz-opendata/
var csvfile = "./23-kz-park_utf8.csv";
d3.csv(csvfile,function(csv) {
        points = csv;
        svg = d3.select("svg"); // <svg>を選択

        svg.selectAll(".node")      //  <circle class="node">を選択
                .data(points).enter()       // データの増分を対象
                .append("circle")           // svgのcircleを追加
                .attr("class","node")
                .attr("cx",function(d) { return projection([d.経度,d.緯度])[0];})    // 円の中心
                .attr("cy",function(d) { return projection([d.経度,d.緯度])[1];})    // 円の中心
                .attr("r",function(d) { return 5;})                                  // 円の半径
                .style("stroke",function(d) { return "#000000";})                    // 枠線の色
                .style("stroke-width","1px")                                         // 枠線の太さ
                .style("fill",function(d) {                                          // 塗りつぶしの色
                        if (d.名称.indexOf("公園")>=0){
                                return "white";
                        } else {
                                return "red";
                        }
                })
                .on('click',function(d) {
                        console.log(d);
                        address = [{Name:d.名称,緯度:d.緯度,経度:d.経度,}];
                        svg.selectAll(".address")
```

課題ー７：step7.js を改造し、「地域防災拠点」を描画

```
35      ////////////////////////////////////////////////////////////////////
36      // 金沢区内公園 http://www.city.yokohama.lg.jp/kanazawa/kz-opendata/
37      var csvfile = "./23-kz-park_utf8.csv";
38      d3.csv(csvfile, function(csv) {
39          points = csv;
40          svg = d3.select("svg");  // <svg>を選択
41
42          svg.selectAll(".node")      //  <circle class="node">を選択
43              .data(points).enter()       // データの増分を対象
44              .append("circle")           // svgのcircleを追加
45              .attr("class", "node")
46              .attr("cx", function(d) { return projection([d.経度, d.緯度])[0]; })    // 円の中心
47              .attr("cy", function(d) { return projection([d.経度, d.緯度])[1]; })    // 円の中心
48              .attr("r", function(d) { return 5; })                                  // 円の半径
49              .style("stroke", function(d) { return "#000000"; }) // 枠線の色
50              .style("stroke-width", "1px")                       // 枠線の太さ
51              .style("fill", function(d) {                        // 塗りつぶしの色
52                  if (d.名称.indexOf("公園")>=0){
53                      return "white";
54                  } else {
55                      return "red";
56                  }
57              })
58              .on('click', function(d) {
59                  console.log(d);
60                  address = [{Name:d.名称, 緯度:d.緯度, 経度:d.経度, }];
61                  svg.selectAll(".address")
62                      .data(address)
63                      .attr("x",function(d) { return projection([d.経度, d.緯度])[0]; })
64                      .attr("y",function(d) { return projection([d.経度, d.緯度])[1]; })
65                      .text(function(d) { return d.Name; })
66                  ;
67              });
68          ;
69
70          // 住所を表示する入れ物を作っておく
71          address = [{Name:"ダミー", }];
72          svg.selectAll(".address")
73              .data(address).enter()
74              .append("text")
75              .attr("class", "address")
76              .attr("x",function(d) { return 0; })
77              .attr("y",function(d) { return 0; })
78              .style("font", "8pt sans-serif")
79              .text(function(d) { return d.Name; })
80          ;
81      });
82  });
83  }
84
```

ファイル名を「./shelter_utf8.csv」にし、
csvファイルを同ディレクトリにコピー

csvのタイトルに合わせる

「小学校」は白、それ以外は赤

csvのタイトルに合わせる

余裕のある人用の、機能追加案

- Nameによって、色を変える
  - 小学校、中学校、その他
- 他の情報も合わせて表示
  - 津波非難施設、応急給水拠点、帰宅困難者一時滞在施設
- アニメーション
  - 施設名の表示を工夫する
    - .transition().duration(ミリ秒).変化後のスタイル()
  - JavaScriptのsetInterval()の利用
    - 静岡県雨量を使った例
      - http://cloud.aitc.jp/20140627_D3js/sample6.html
    - テレビ放送受信契約数を使った例
      - http://cloud.aitc.jp/20140717_D3js/tv/step6example.html
- 地図上に、自分の現在位置を表示
  - HTML5のGeolocation API

今後の拡張の方向

- 付加情報を表示
  - clickかmouseoverで、詳細情報を表示
    - 地名・住所・関連情報など
  - 文字をもっと見易くする（白抜き文字、など）
    - http://www.slideshare.net/kadoppe/inline-svg/53
- 他のデータも合わせて表示
  - 他のオープンデータ
  - 社内の業務データなど
- アニメーション
  - 「もう１つ軸を足したい」時など
- 拡大／縮小
  - 特定の区を詳細に見たい
  - 倍率に合わせて、円の大きさや文字サイズを調整

注意事項

- 元データが変な形式だった場合
  - テキストエディタやExcelで変換した方が簡単
    - 「読み込めばOK」な状態にしておく
  - 複数のファイルになっても問題なし
  - RDBのテーブル設計のセンスがあれば、完璧

- データのライセンスに注意
  - http://www.city.yokohama.lg.jp/front/aboutweb.html

  > 私的使用のための複製や引用など著作権法上認められた場合を
  > 除き、無断で複製・転用をすることはできません。

  横浜市に勉強会での使用について問い合わせた結果
  > 「横浜市統計書」からの出典を明らかにしていただければ、
  > 著作権法上認められた場合に該当すると判断します。

自分の興味あるデータを探して、描画してみる ～17:45

ネット環境の無い人は、
sampleの下のデータを使ってください。

17:45～ 発表タイム

発表タイム

- どんなものが出来ましたか？