D3.js入門

西田 直樹

はじめに

- 必要なもの: モダンなブラウザ
  - ChromeかFirefoxが無難
  - Opera…?

- スライドとサンプルコードはすべてGitHubにアップロードしてあります
  - https://github.com/domitry/KC3_D3js
- 手元でスライドを見ながら進めてください.

はじめに

- d3.jsはかなり巨大なライブラリなので全てを説明はしません。適宜自分で調べることをお勧めします。
- d3.jsのGitHub wikiが一番情報量が多いのでおすすめです。(https://github.com/mbostock/d3/wiki)
- Tutorialもお勧め(http://ja.d3js.info/alignedleft/tutorials/d3)

自己紹介

- 西田 直樹 @domitry
- 専門: 生物物理
- A member of SciRuby, E-cell project
- Simulation, Visualization

Agenda

- D3.jsとは
- D3.jsができること
- JavaScriptの基本
- D3.jsの基本
- やってみよう
- 終わり

D3.jsとは

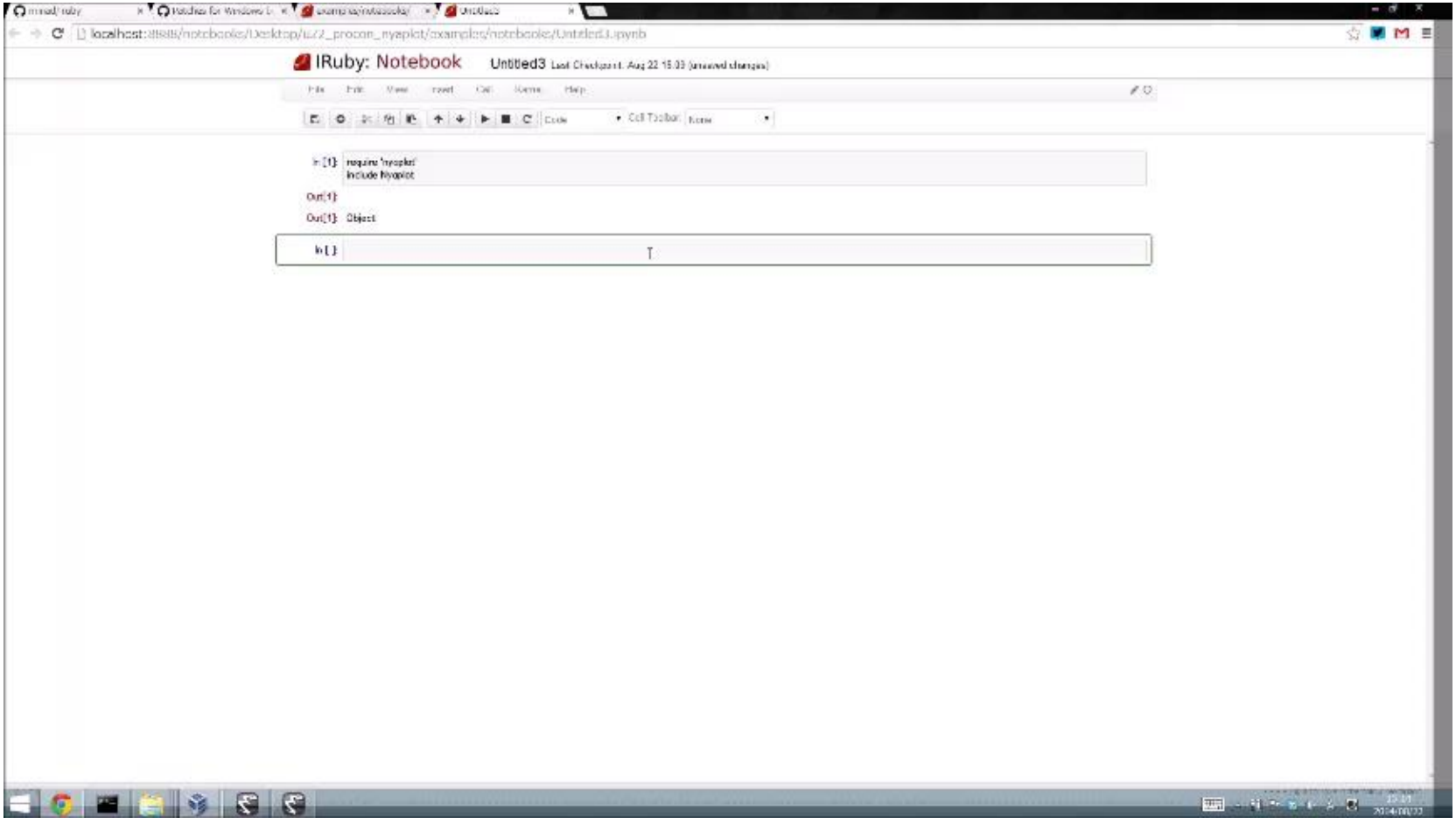
- Data Driven Documents -> d3
- データ可視化のためのナウでヤングなライブラリ
- 詳しくは http://d3js.org/

D3.jsができること

- 色々。jQueryの代替のようなことから高レベルなSVGの構築など
- 統計な人たちからBioinformatics屋さんまで
- 他言語との連携が熱い

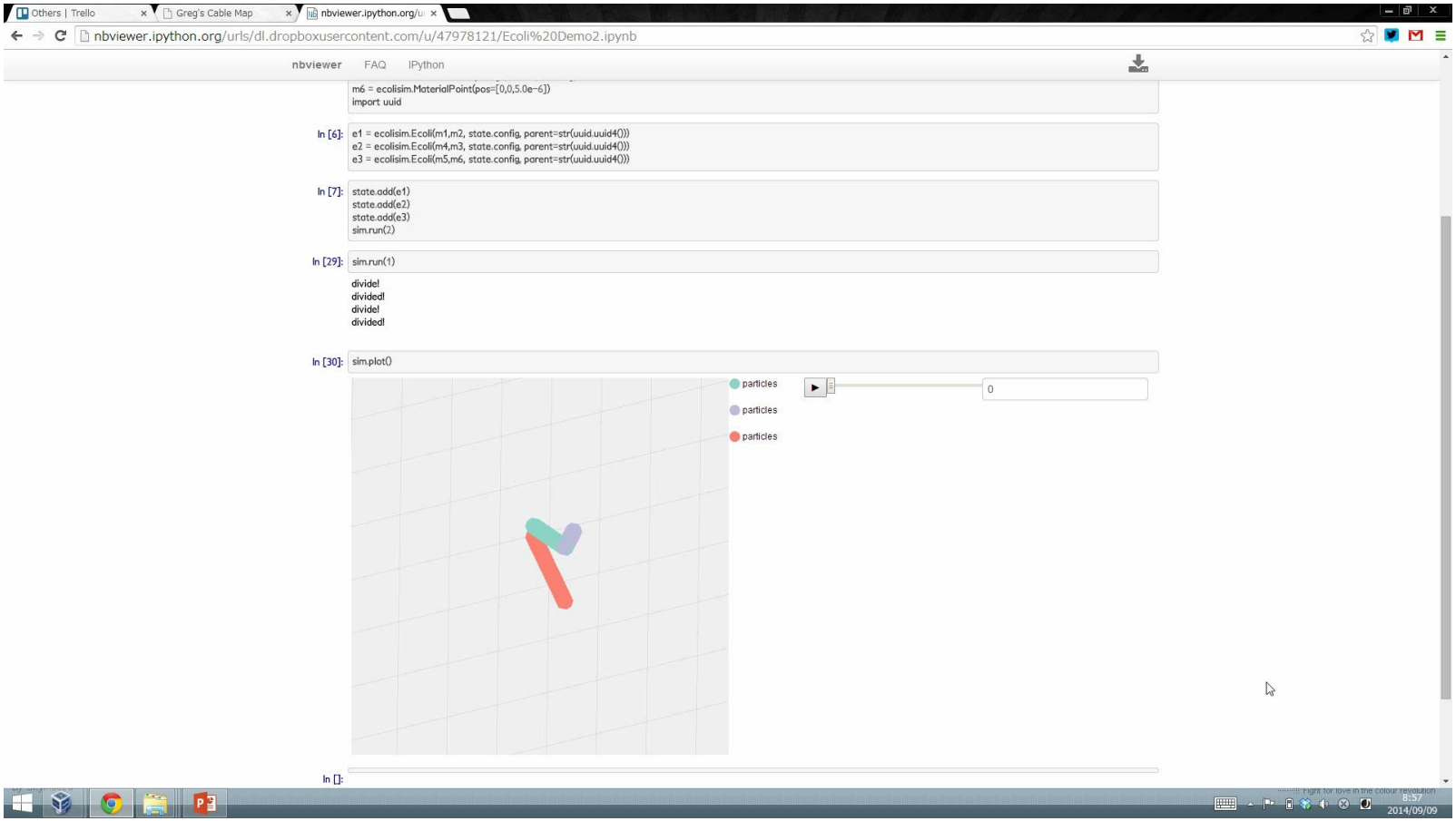
Nyaplot

- D3.js+Ruby



Elegans

- D3.js+WebGL



JavaScriptの基本

Question

- プログラミングしたことある人?

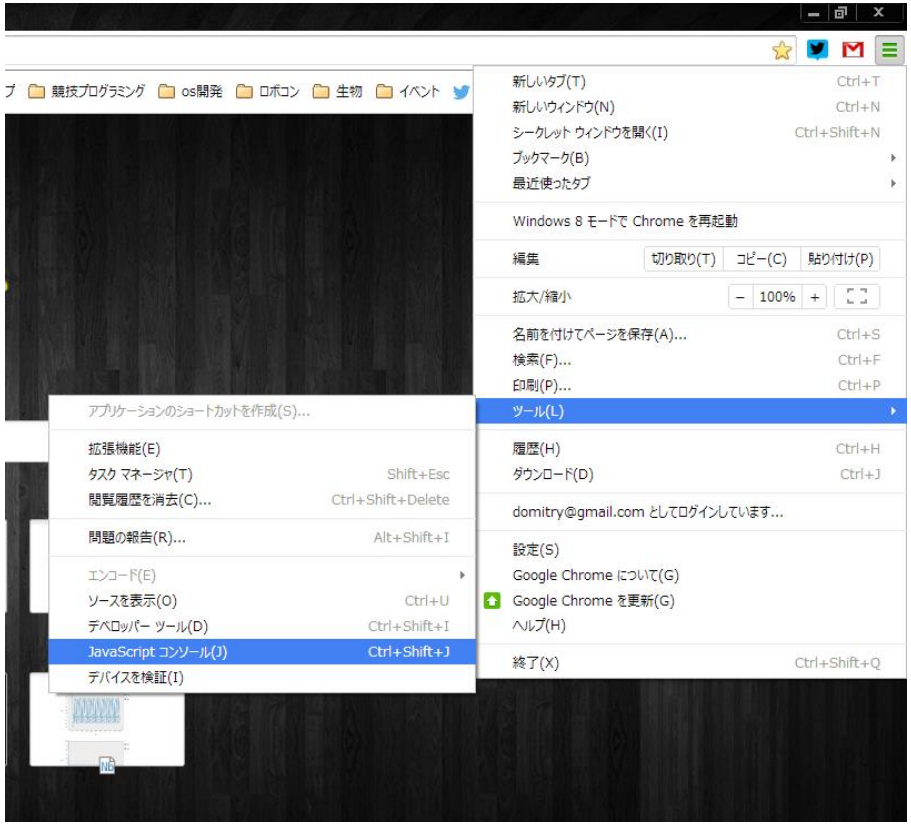
Question

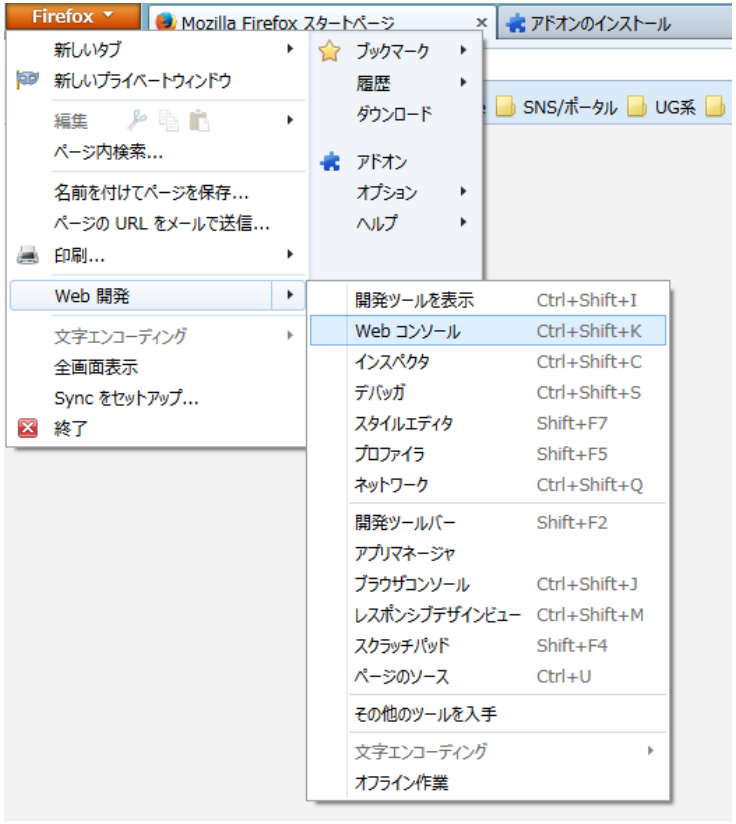
- プログラミングしたことある人?
- C言語書いたことある人?

Question

- プログラミングしたことある人?
- C言語書いたことある人?
- JavaScript書いたことある人?

JavaScriptコンソール





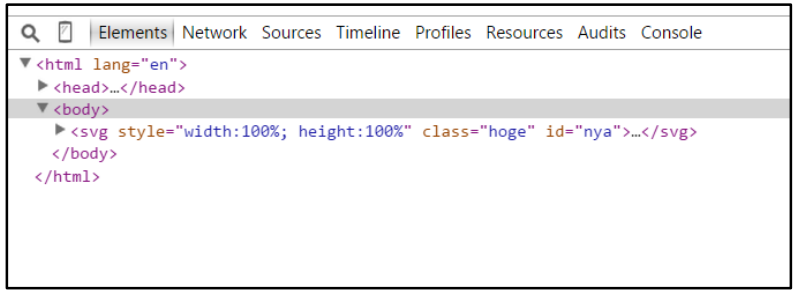
JavaScriptコンソール

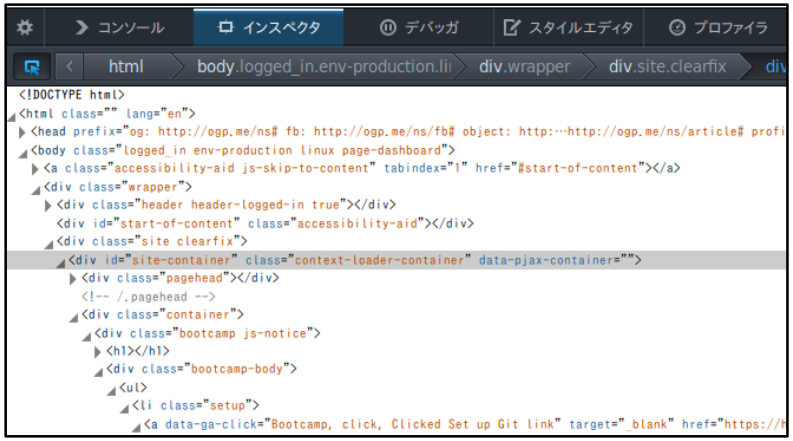
- コードを入力するとその場で実行
- Shift+Enterで複数行入力

```
console.log( “Hello, world!” );
```

インスペクタ

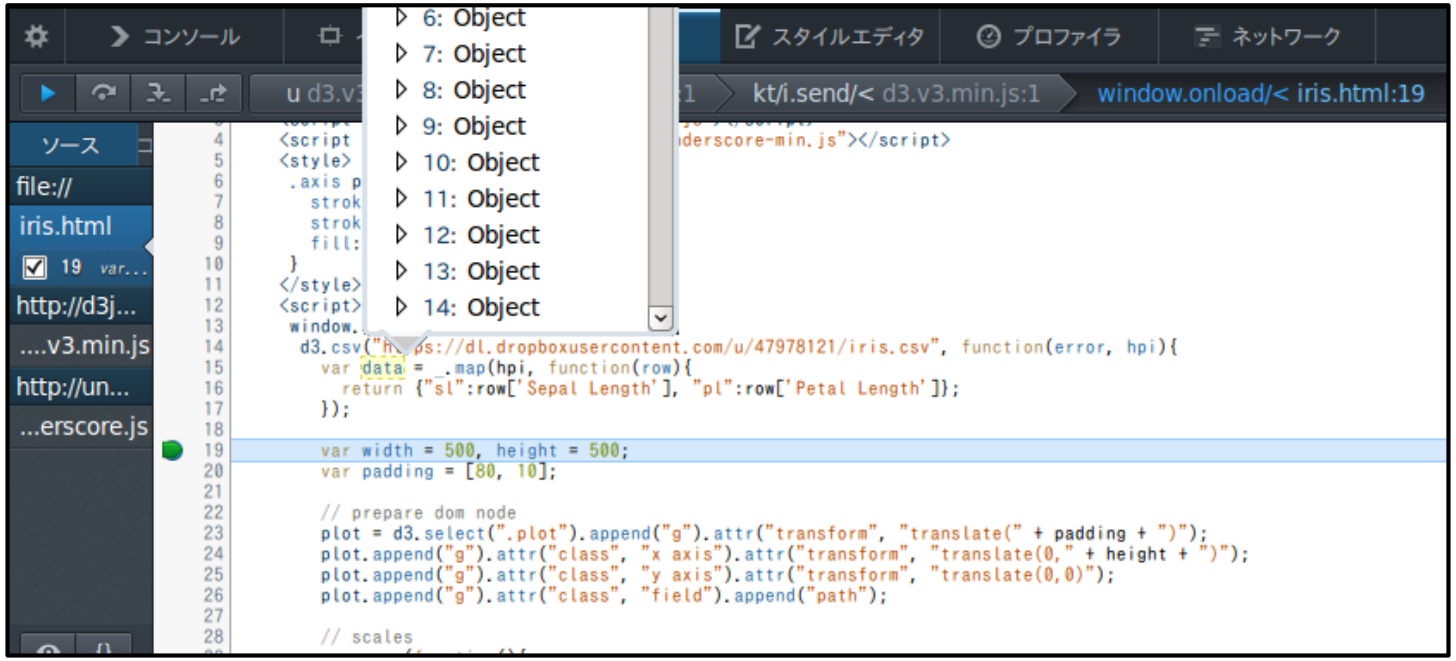
- DOMツリーを見ることができる
- DOM要素が動的に追加されるときに便利





デバッガ

- ブレイクポイントを挟んでその時々の変数の中身を確認できる。



変数と型

- JavaScriptは動的型付け言語
  - Number -> intやfloatの区別なし
  - Array
  - Object　-> 実は割と皆Object, 今回はHash的に使う
  - Function

```
typeof {a: 3, b: 4, c: 10} //->?
typeof [10, 20, 30] //->?
```

制御文の数々

- 基本的にCと一緒！(と思っていたら痛い目にあうこともある)
  - if
  - switch, case
  - for
  - while
  - break, continue

関数

- functionで定義
- 引数の指定に型はいらない

```
function Str2Int(str){
        return ParseInt(str);
}

Str2Num( “3” ); //-> 3
```

無名関数・クロージャ

- window.onloadに無名関数を代入するとページのロードが終わった時点で実行される。

```
window.onload = function(){
        console.log( “Hello, world!” );
};
```

無名関数・クロージャ

- 先ほどのStr2Intは次のように書ける。
- 関数はFunction型のオブジェクト

```
var Str2Int = function(str){
        return ParseInt(str);
};

Str2Num( “3” ); //-> 3
console.log(typoeof Str2Int); //->  ‘function’
```

無名関数・クロージャ

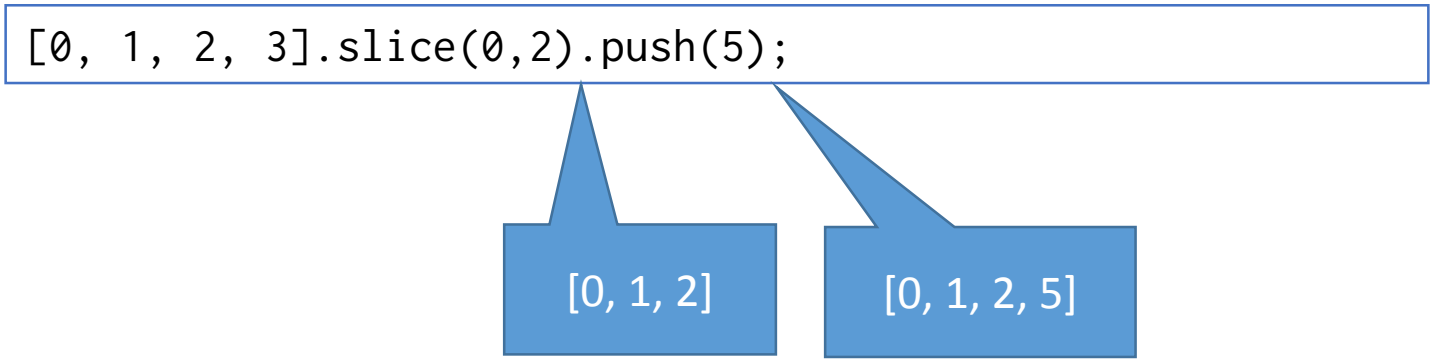
- スコープを汚したくないときは次のようにも書く

```
var hoge = (function(){
        var a = 5;
        var b = 3;
        return a*b;
})();

console.log(typeof a); //->  ‘undefined’
```

メソッドチェーン

- メソッドの返り値が同じオブジェクト，メソッドを次々とつなげていく
- 下の例ではsliceの返り値がArray



D3.jsの基本

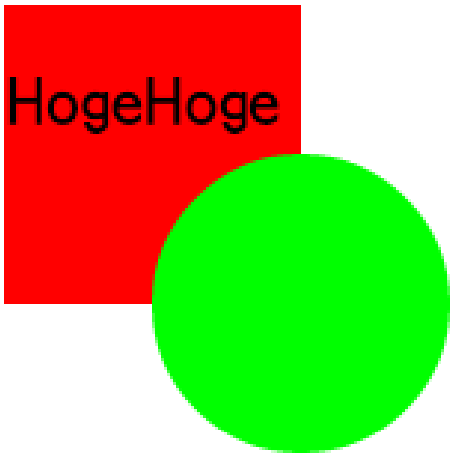
D3.jsの基本

- ここで解説するもの
  - selector
  - data, datum
  - scale

- サンプルコードは下記URLを開いてJavaScriptコンソールで実行してください
  - http://goo.gl/DDwji8

Inline SVG

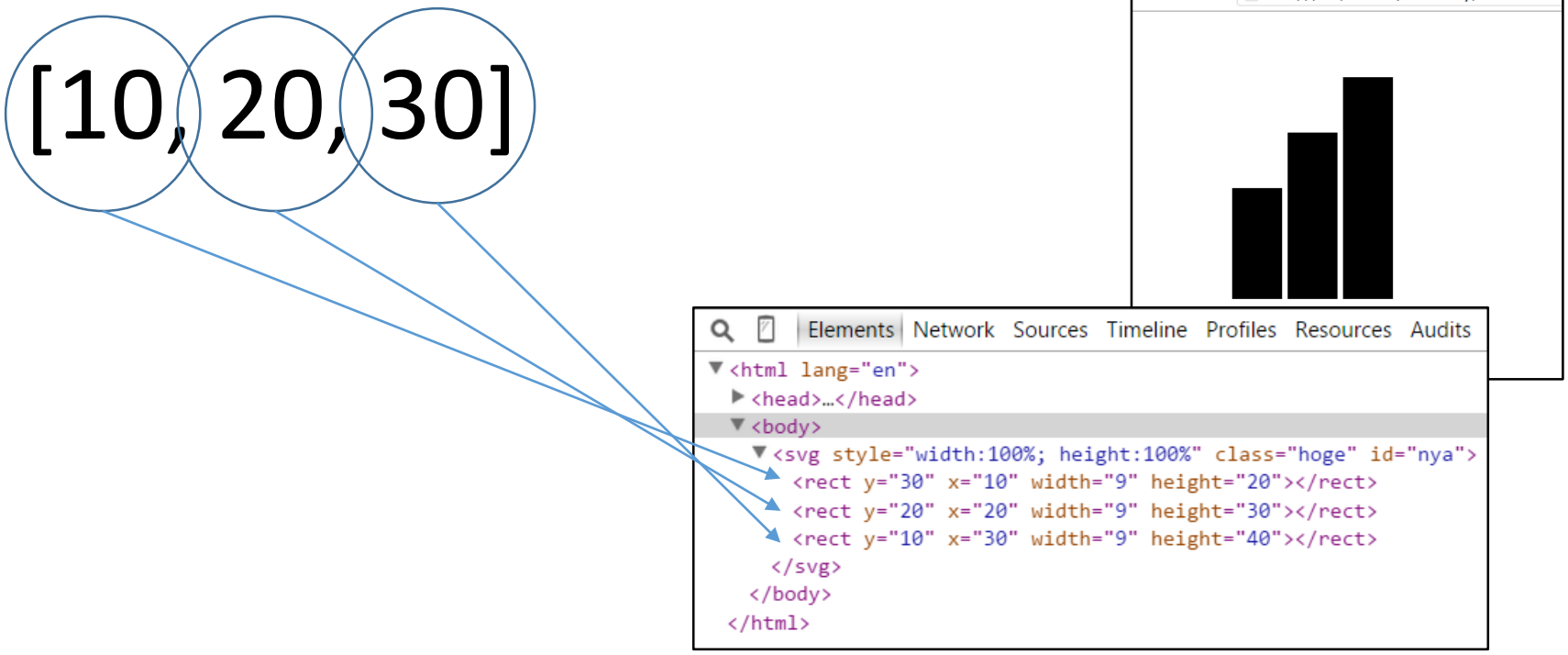
- htmlの中に記述できる
- 図形(circle, rect, etc.)をDOM要素として定義




```
<html lang="en">
  <head>
    <script src="http://d3js.org/d3.v3.min.js"></script>
    <script src="http://underscorejs.org/underscore-min.js"></script>
  </head>
  <body>
    <svg style="width:100%; height:100%" class="hoge" id="nya">
      <rect x=0 y=0 width=100 height=100 fill="#f00"></rect>
      <circle cx=100 cy=100 r=50 fill="#0f0"></circle>
      <text x=0 y=40 font-size=22>HogeHoge</text>
    </svg>
  </body>
</html>
```

D3.jsの考え方

- データをDOM要素にバインドする
- バインドされたデータに基づいてDOMの属性(width, height, etc.)を変える



DOM操作の流れ

- コンソールで入力してみてください。

```
d3.select( “svg” ).remove();
d3.select( “body” ).append( “svg” );
var rects = d3.select( “svg” ).selectAll( “rect” ).data([1, 10,
100]).enter();
rects.append( “rect” ).attr({
        x: function(d){return d/10;},
        y: 0,
        width: function(d){return d;},
        height: function(d){return d;}
});
```

DOM操作の流れ

- svg要素を子供ごと捨ててbodyに新しいsvgをくっつける

```
d3.select( “svg” ).remove();
d3.select( “body” ).append( “svg” );
var rects = d3.select( “svg” ).selectAll( “rect” ).data([1, 10, 100]).enter();
rects.append( “rect” ).attr({
        x: function(d){return d/10;},
        y: 0,
        width: function(d){return d;},
        height: function(d){return d;}
});
```

DOM操作の流れ

- svg要素の子供のrect3つにそれぞれ1, 10, 100をバインドする

```
d3.select( “svg” ).remove();
d3.select( “body” ).append( “svg” );
var rects = d3.select( “svg” ).selectAll( “rect” ).data([1, 10, 100]).enter();
rects.append( “rect” ).attr({
        x: function(d){return d/10;},
        y: 0,
        width: function(d){return d;},
        height: function(d){return d;}
});
```

rectsを覗いてみる

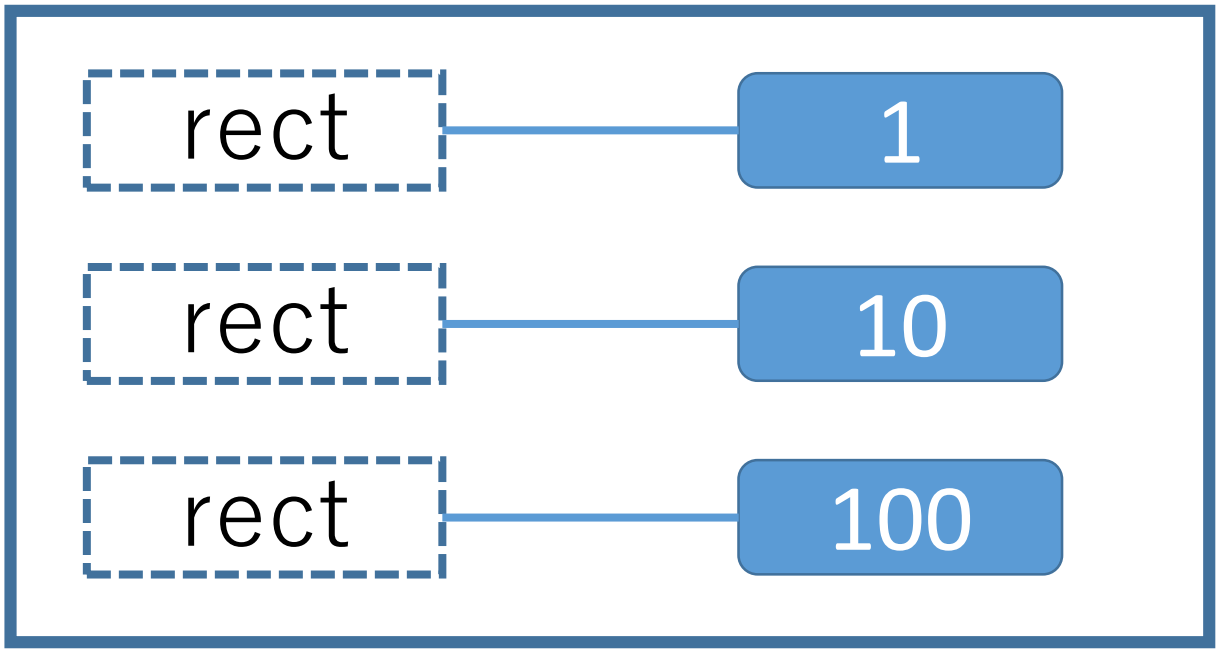
- rectsはArray, 要素のプロパティの__data__にバインドしたデータが入っている
- それぞれの要素は”selection”

Selection

- DOMの入れ物。
- まだ存在していない要素をselectすると入れ物だけ返ってくる。
- enter()で新しくデータがひも付けされたselectionのみを選択

```
d3.select( “svg” ).remove();
d3.select( “body” ).append( “svg” );
var rects = d3.select( “svg” ).selectAll( “rect” ).data([1, 10,
100]).enter();
```

rects: イメージ




```
var rects = d3.select( “svg” ).selectAll( “rect” ).data([1, 10,
100]).enter();
```

selectorのメソッド

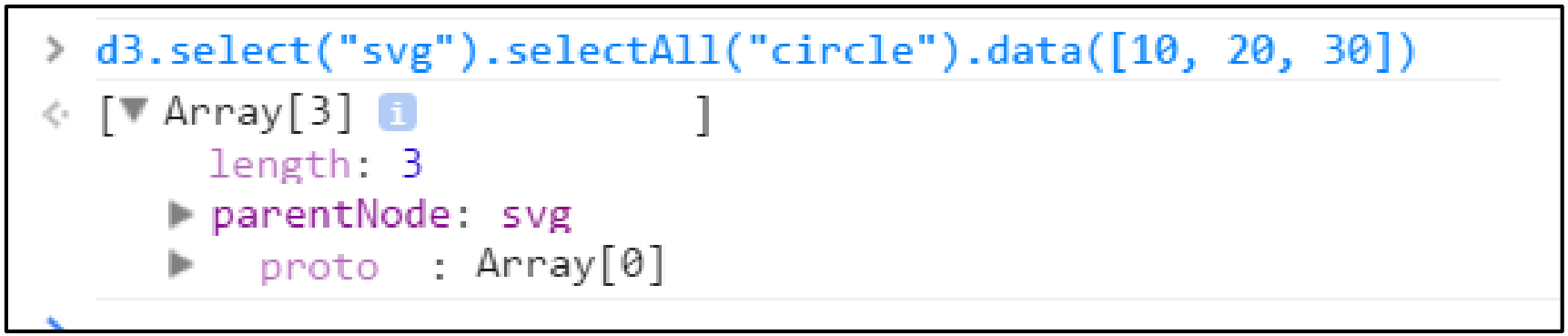
- appendでDOM要素を生成, attrで属性を設定。
- attrの引数に関数を与えることでバインドしたデータを参照できる
- attr(“x”, function(d){}).attr(“y”, function(d){}) のようにも書ける

```
d3.select( “svg” ).remove();
d3.select( “body” ).append( “svg” );
var rects = d3.select( “svg” ).selectAll( “rect” ).data([1, 10,
100]).enter();
rects.append( “rect” ).attr({
        x: function(d){return d/10;},
        y: 0,
        width: function(d){return d;},
        height: function(d){return d;}
});
```

ややこしいところ

- enter()?
- enterしないで見てみる

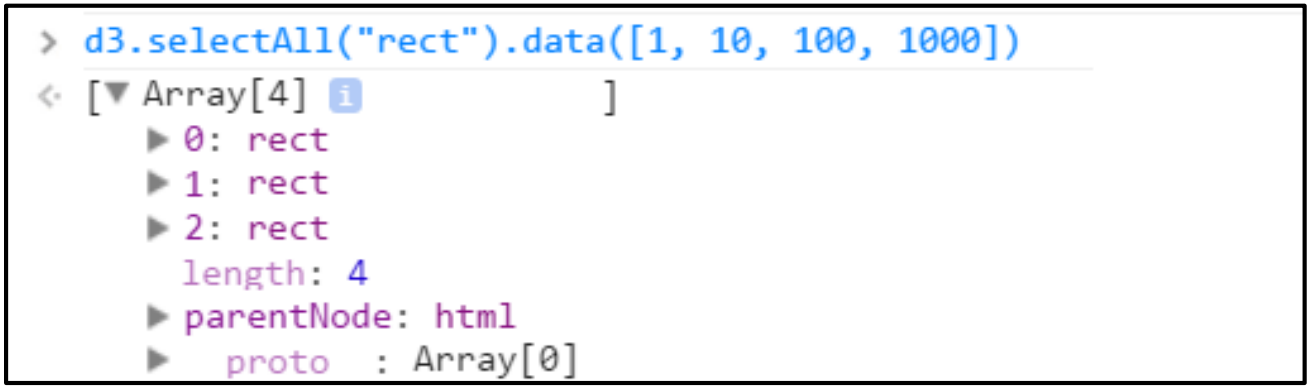
    -> selectorが生成されてない



ややこしいところ

- rectの要素を増やしてみる

  -> 増えた要素の分だけ生成されていない



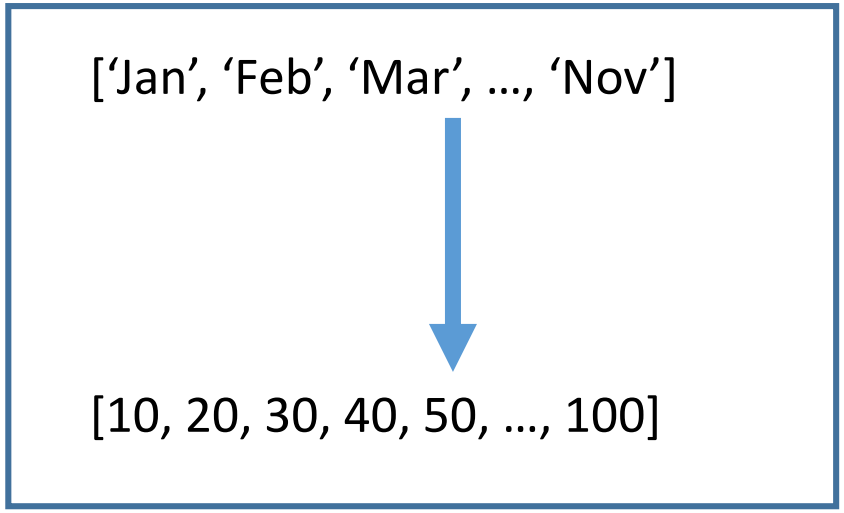
enterとは

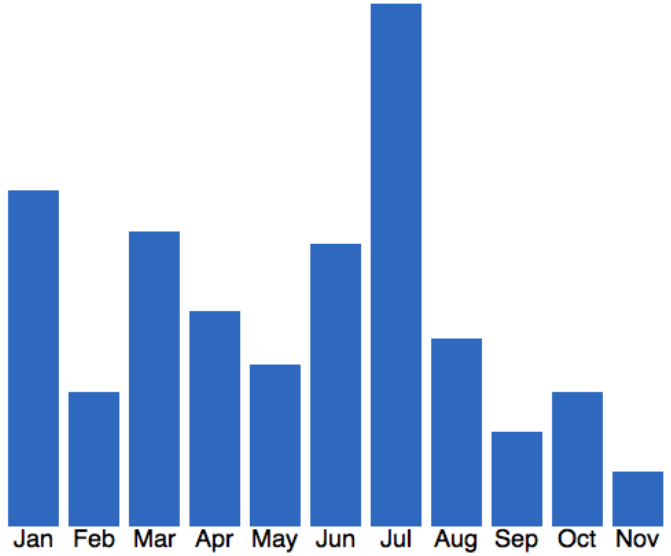
- 足りない要素のselectionを生成、選択してグループに入れるメソッド(selection)
- 逆に元より少ないデータをバインドした後exit()を呼び出すと余った要素が選択される
- “三つの子円”(http://ja.d3js.node.ws/document/tutorial/circle.html)

閑話休題

scale

- 便利メソッドのひとつ
- データから実際の座標に変換





scale

- d3.scale.linear()
  - domain – 入力データ
  - range – 出力座標

```
var scale = d3.scale.linear().domain([0, 2]).range([0, 100]);
scale(1);  //-> 50
scale(2);  //-> 100
scale(1.5);//-> 75
```

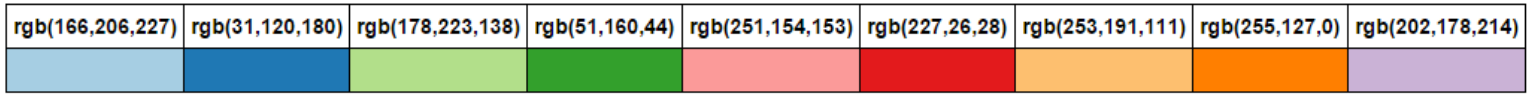
scale

- データの種類によって色々なscaleが定義
  - linear – 連続データ, 線形対応
  - log – 連続データ, 対数目盛のイメージ
  - ordinal – 離散データ

```
var scale = d3.scale.ordianl().domain([ “hoge” , “nya” ,
 “nyaa” ]).range([0, 1, 2]);
scale( “hoge” );  //-> 0
scale( “nya” );   //-> 1
scale( “nyaa” );  //-> 2
```

scale

- 座標だけでなく色もrangeに指定できる




```
var scale = d3.scale.linear().domain([0, 100]).range([ “#fff” ,
 “#f00” ]);
scale(30);   //-> "#ffb3b3"
scale(50);   //-> "#ff8080"

var scale = d3.scale.category10();
scale( “hoge” );    //-> "#1f77b4"
scale( “nya” );     //-> "#ff7f0e"
scale( “hoge” )     //-> "#1f77b4"
```

Underscore.jsの基本(おまけ)

_.map

- 第二引数のfunctionに要素を一つ一つ代入, 返り値を配列にまとめて返す

```
_.map([0, 1, 2], function(val){
        return val*2;
}); // -> [0, 2, 4]
```

_.reduce

- valには左から順番に要素の値が入る
- memoは最初は{}, 次からfunctionの返り値
  - {}, {‘1’:10}, {‘1’:10, ‘2’:9}

```
_.reduce([10, 9, 6], function(memo, val, i){
        memo[String(i)] = val;
        return memo;
}, {}); // -> { ‘1’ :10,  ‘2’ :9,  ‘3’ :6}
```

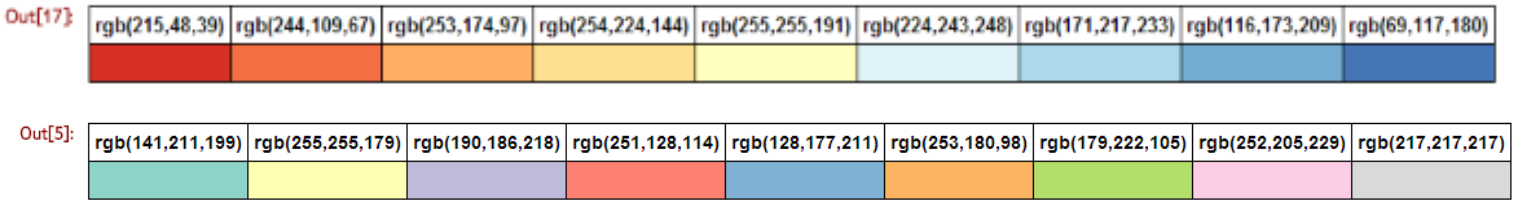
便利関数色々

- max, min, each, isUndefined, isArray, etc.
- 詳しくはhttp://underscorejs.org/

補足

Colorbrewer

- 用途別のカラーセット
- http://colorbrewer2.org/
- The Apache License



やってみよう

Attention

- templates/* にテンプレートがあります
- answers/* に解答例があります。
- 好きなものからやってください。(最初の2つは解説します)

Assignments#0

- templates/iris.html
- answers/iris.html
- 散布図の練習

Assignments#0

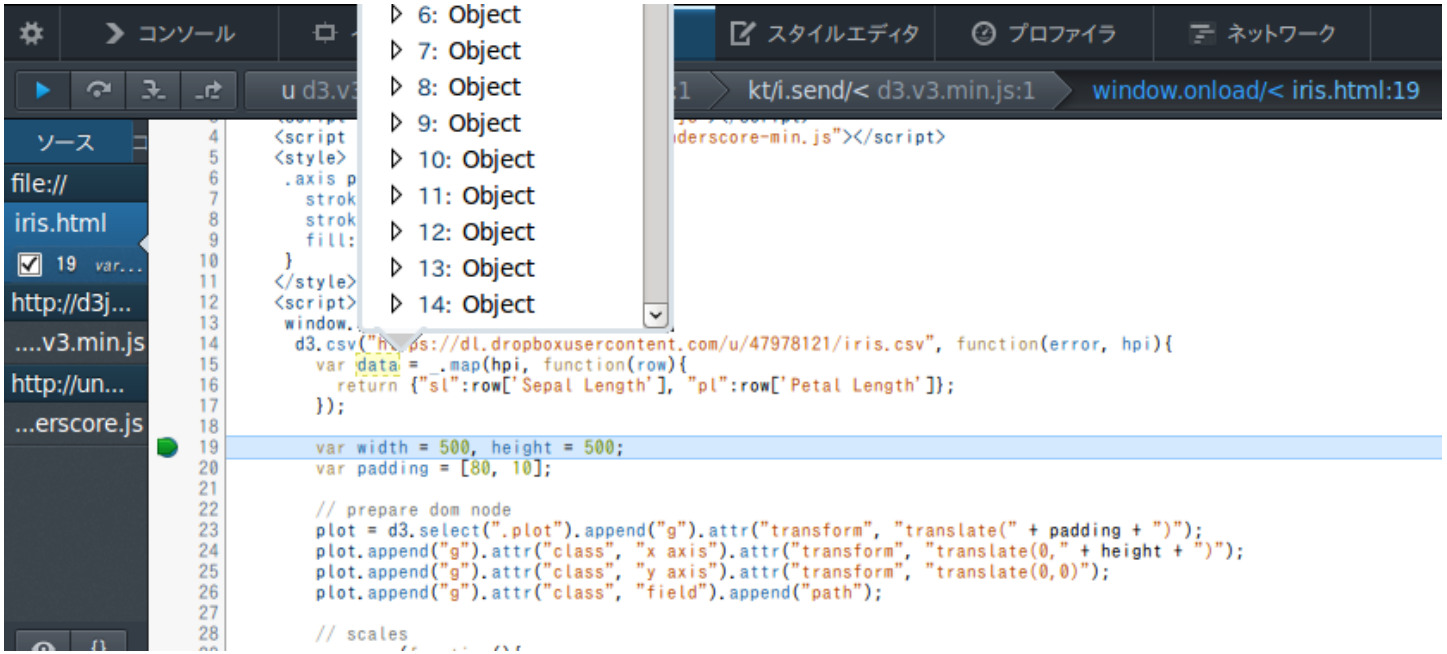
- iris.csv
  - 言わずと知れた超有名データセット
  - あやめ(iris)の花びら(Petal)の長さと, がく片(Sepal)の長さに相関があるか調べる

Assignments#-1

- やること
  - データを解析してscaleを作る(★☆☆)
  - 散布図(scatter)を作る(★☆☆)
  - 種(Speciesごとに)色分けをする(★☆☆)
  - 軸にラベルをつける(★☆☆)

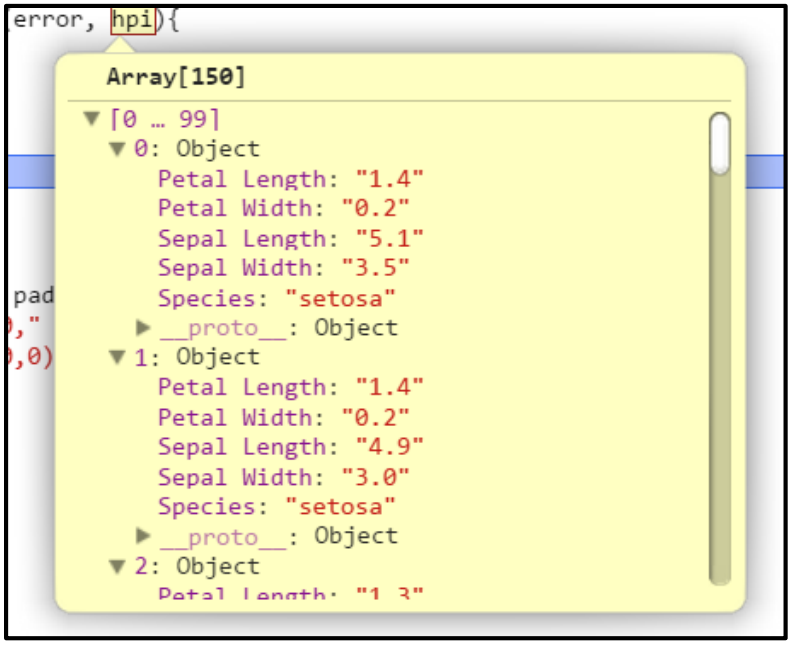
Assignments#-1 scaleを作る

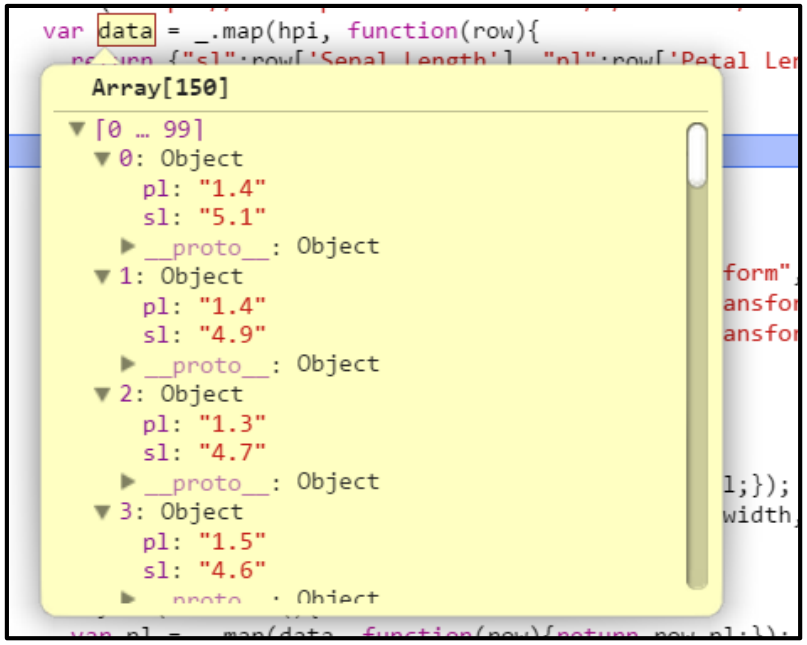
- templates/iris.htmlの19行目にブレイクポイント
- 元のデータ(hpi)と加工後のデータ(data)が見える



Assignments#-1 scaleを作る

- 必要なデータだけdataへ移動





Assignments#-1 scaleを作る

- やること
  - domain: データセットの範囲
  - range: 画面の範囲
- 下図のslはがく片の長さの配列

```
// scales
var x = (function(){
  var sl = _.map(data, function(row){return row.sl;});
  return d3.scale.linear().domain([0, 1]).range([width, 0]);
})();
```

Assignments#-1 scaleを作る

- 使える関数：_.min, _.max

```
_.min([0,1,2]); //-> 0
_.max([0,1,2]); //-> 2
```

- できたらコンソールを閉じてリロードしてみる

Assignments#-1 散布図を作る

- インスペクタを開く
  - class属性がfieldのg要素(group)にcircleを追加すればよい

Assignments#-1 散布図を作る

- class属性がfieldのg要素にcircleを追加
- cxとcyを決めるには先ほど作ったscaleを使う

```
d3.select( “.field” )
        .selectAll( “circle” )
        .data(data)
        .enter()
        .append( “circle” )
        .attr({
                ‘cx’ : ?,   // 中心のx座標
                ‘cy’ : ?,   // 同y座標
                ‘r’  : 3,   // 半径
                ‘fill’ : ” #000”
        });
```

Assignments#-1 散布図を作る

- dには先ほどバインドしたdataの要素一つ一つが入る
  - 例えば {pl: 1.4, sl: 2,5}
  - d.pl, d.slでアクセス

```
d3.select( “.field” )
        .selectAll( “circle” )
        .data(data)
        .enter()
        .append( “circle” )
        .attr({
                ‘cx’ : function(d){ ? },
                ‘cy’ : function(d){ ? },
                ‘r’  : 3,   // 半径
                ‘fill’ : ” #000”
        });
```

Assignments#-1 散布図を作る

- リロードしてみる

```
d3.select( “.field” )
        .selectAll( “circle” )
        .data(data)
        .enter()
        .append( “circle” )
        .attr({
                ‘cx’ : function(d){return x(d.pl);},
                ‘cy’ : function(d){return y(d.sl);},
                ‘r’  : 3,
                ‘fill’ : ” #000”
        });
```

Assignments#-1 色分けをする

- Speciesで色分けをしたい。
- color scaleを用意(今回は組み込みのcategory10を使う)

```
var color_scale = d3.scale.category10();
```

Assignments#-1 色分けをする

- 16行目, 元のデータからSpeciesの情報を抜き出してdataに入れておく

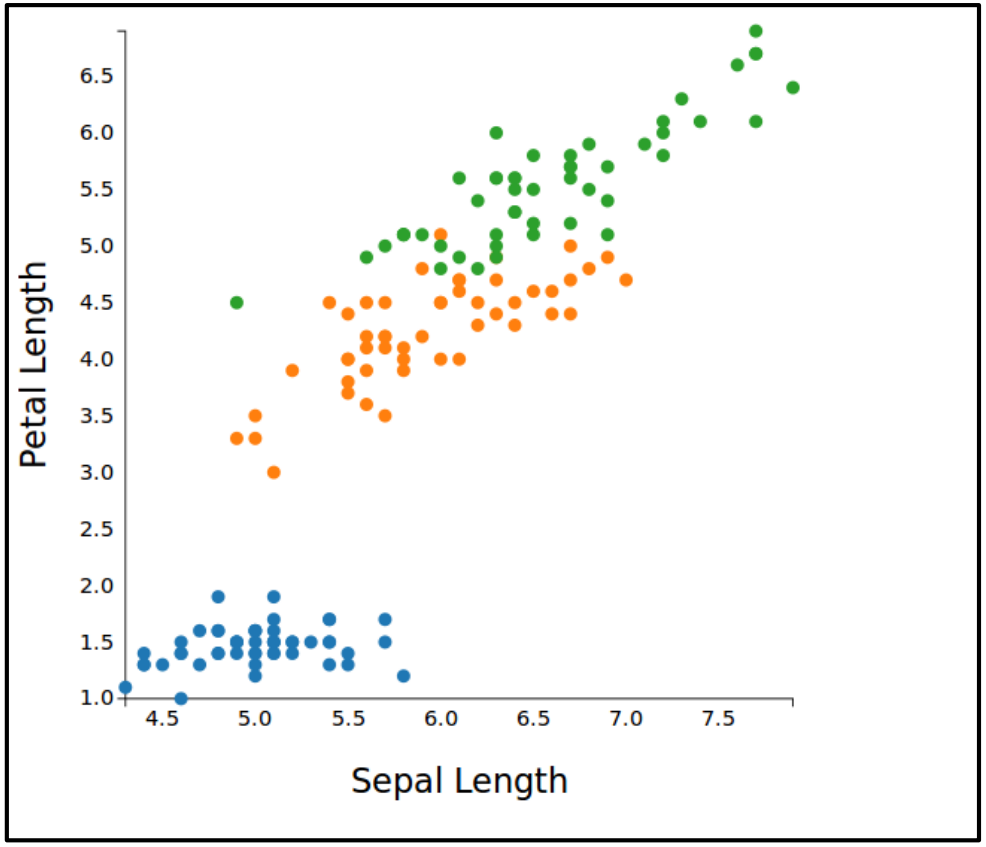
```
return { “sl”:row['Sepal Length'], "pl":row['Petal Length']
 “species” : ?};
```

Assignments#-1 色分けをする

- fill属性を変える.

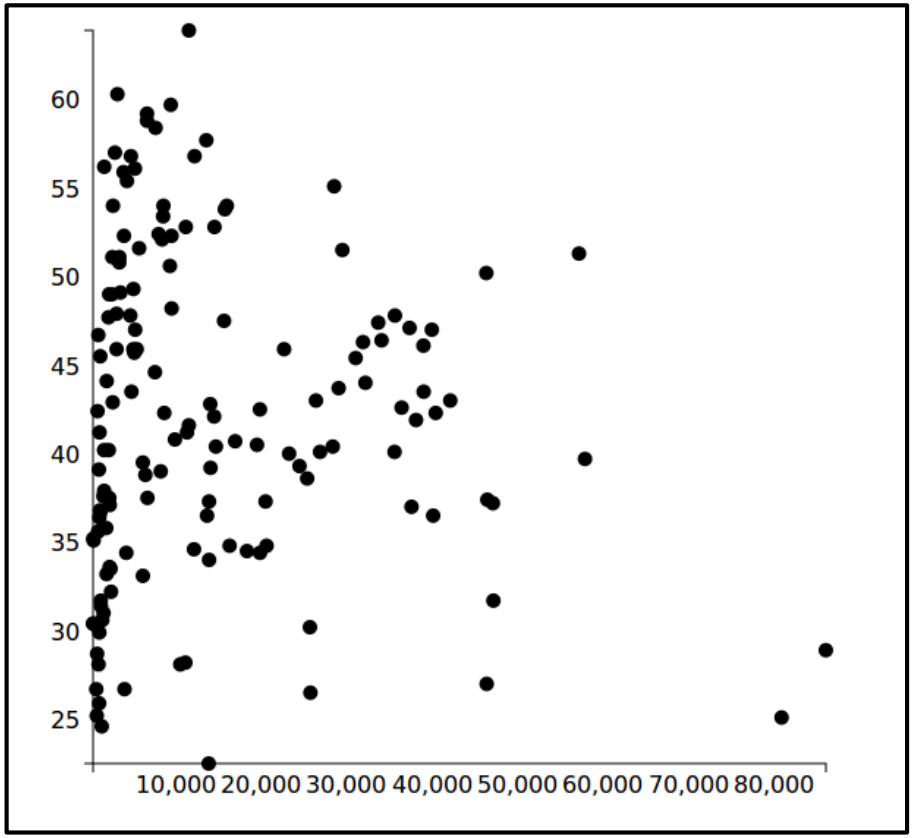
```
d3.select( “.field” )
        .selectAll( “circle” )
        .data(data)
        .enter()
        .append( “circle” )
        .attr({
                ‘cx’ :function(d){return x(d.sl);},
                ‘cy’ :function(d){return y(d.pl);},
                ‘r’  :5,
                ‘fill’ :function(d){ ? }
        });
```

Questions?


Petal Length
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
Sepal Length

Assignments#0

- hpi.html



Assignments#0

- hpi.csv
  - Happy Planet Index 地球幸福度指数
  - これからは経済力で競う時代じゃない, 大事なのは国民の幸福度だ! の指標の一つ.
  - GDPと人口(Population), HPIに相関がありそうか見てみる.
  - 出展: http://www.happyplanetindex.org/

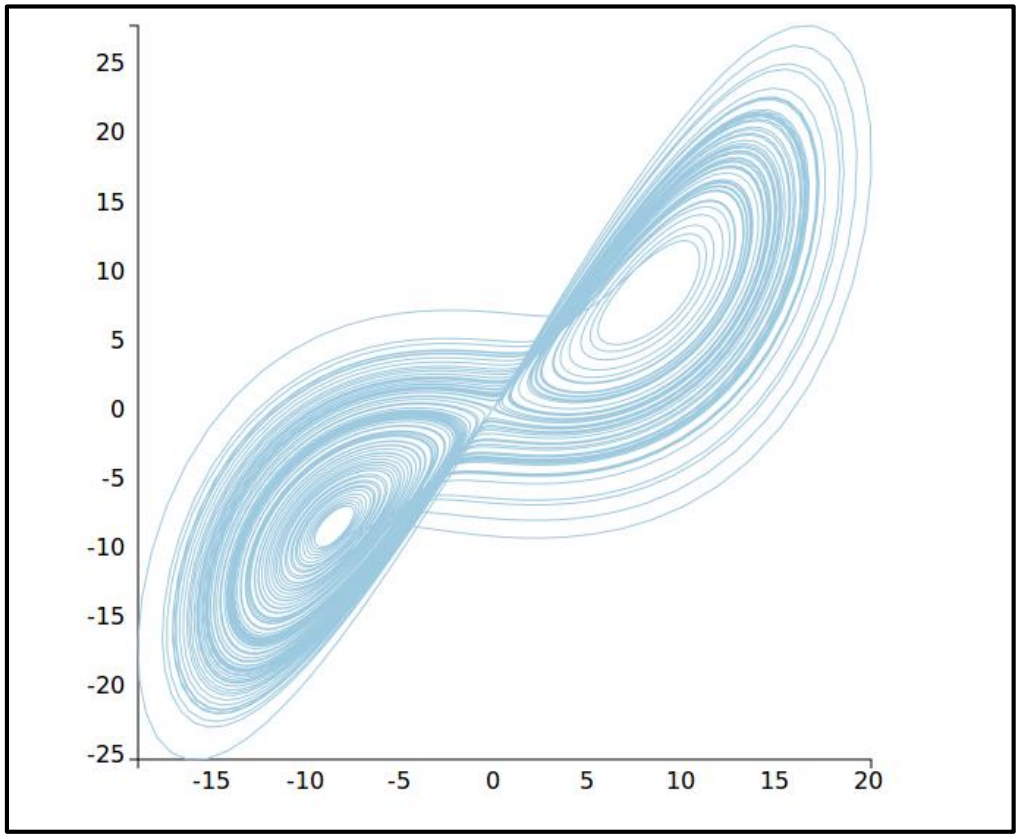
Assignments#0

- やること
  - Assignment#-1と同じように散布図を作成してみる。(☆☆☆)
    - GDPを横軸, HPIを縦軸にする。

- やること(Optional)
  - 人口で色分けしてみる。(★★☆)
    - d3.scale.linear()を使う。
    - Assignment#-1と違いちゃんとrangeとdomainを指定しないといけない

Assignments#1

- attractors.html



Assignments#1

- attractors.html
  - 有名なLorenz Attractor
  - オイラー法で微分方程式を解いてプロット

Assignments#1

- やること
  - templates/attractors.htmlを完成させる
- やること(Optional)
  - setIntervalを使って時間によって線が伸びていくようにする(★★☆)
  - tによって線の色を変える(★☆☆)
  - Chuaなど別のアトラクタを実装する(☆☆☆)
  - アトラクタを切り替えられるようにする(★★★)

Assignments#1

- 色々なattractorがある。方程式や色を変えてみるだけでも面白いかも。
  - http://www.bentamari.com/attractors.html
  - http://qiita.com/domitry/items/8ccbc5f2a02004b80a32

Assignments#1

- d3.svg.line()
  - datumと組み合わせて使う
  - datumはdataと違い一つのDOMに配列を紐づけする

```
d3.select( ‘svg’ )
  .append( ‘path’ )
  .datum([{x: 0, y: 200}, {x: 300, y: 500}])
  .attr( “d” , d3.svg.line()
        .x(function(d){return d.x;})
        .y(function(d){return d.y;})
   )
  .attr( “stroke” ,  “#000” )
  .attr( “stroke-width” ,3);
```

Assignments#2

- wind.html

Assignments#2

- wind.csv
  - NASAの全世界風向・風力データを手頃に加工したもの
  - lon, latは緯経度
  - uwnd, vwndの単位はm/s

Assignments#2

- やること
  - templates/wind.htmlに付け足して地図の上に風向きを表す線をつけてみる.(★☆☆)
- やること(Optional)
  - 線の色を風量によって変える(★☆☆)
  - 線の先に矢印をつける等(★★☆)

Assignments#2

- <line>を使う. 属性は以下のよう。
  - x1, y1: 起点
  - x2, y2: 終点

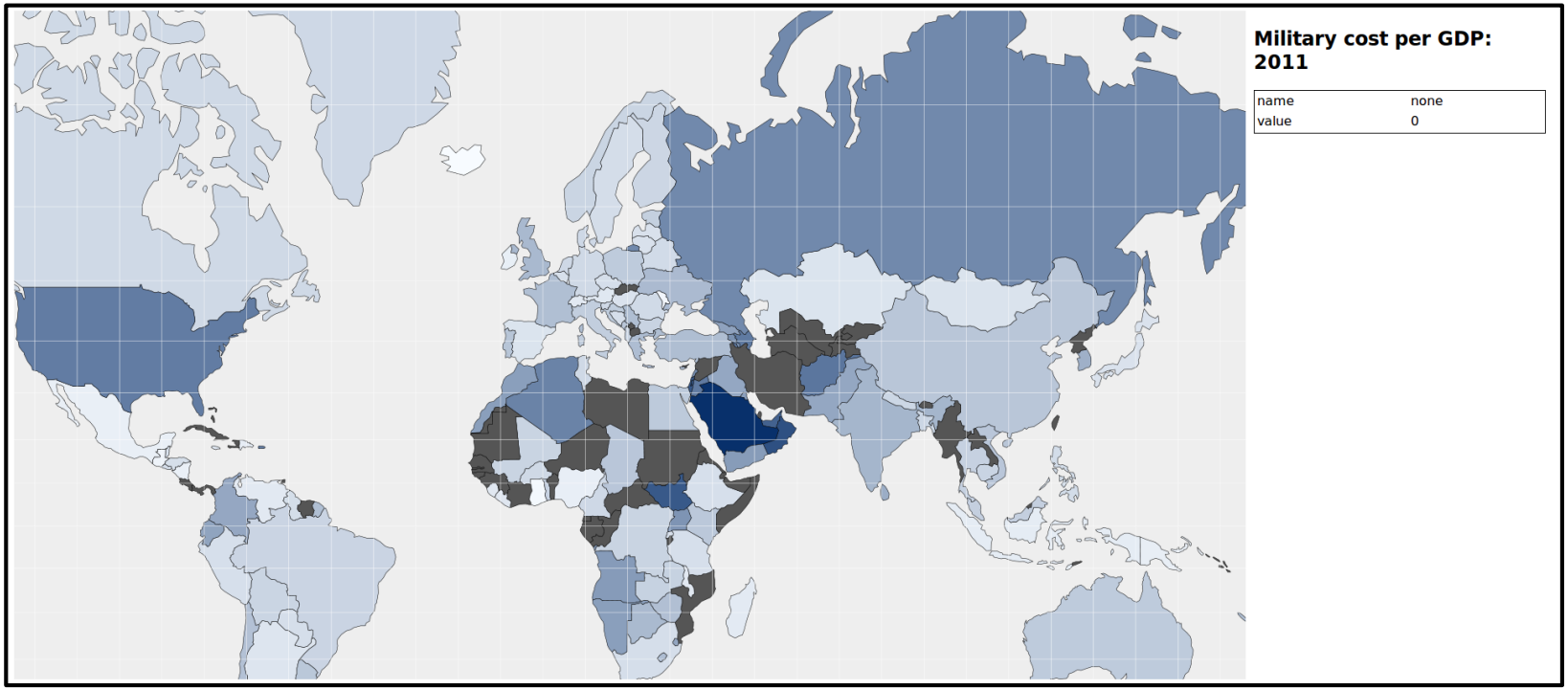
```
d3.select( ‘svg’ )
  .selectAll( ‘line’ ).data(data).enter()
  .append( ‘line’ )
  .attr({
         ‘x1’ :function(d){},
         ‘y1’ :function(d){},
         ‘x2’ :function(d){},
         ‘y2’ :function(d){}
  });
```

Assignments#2

- d3.geo.projection
  - https://github.com/mbostock/d3/wiki/Geo-Projections
  - projection([0, 100])とすると経度0, 緯度100に対応する画面上の座標が[x, y]の配列で返ってくる。
  - windに入っている緯経度を先に画面上の座標に直してからバインドすると楽かも。

Assignments#3

- gdp.html



Assignments#3

- gdp.csv
  - UN dataから, 軍事費の対GDP比を国, 地域別に集計したもの
  - 1980年代から2012年までばらばら
  - 国名もばらばら(できるかぎり修正しました)

Assignments#3

- やること
  - templates/gdp.htmlを改変して軍事費によって地図を塗り分けてみる(★★☆)
- やること(Optional)
  - 左の地図をクリックしたら対応する国名と値を右の表に表示する(★★☆)
  - 表の下に年度による値の推移を折れ線グラフとして表示(★★★)

Assignments#3

- 自分で新しいカラースケールを定義する
  - linear, domainとrangeは?
  - dataを覗いてみる
- 定義したカラースケールを使ってpathを塗り分ける
  - どれを変えたらいいかわからなければ適当に色を変更してみる

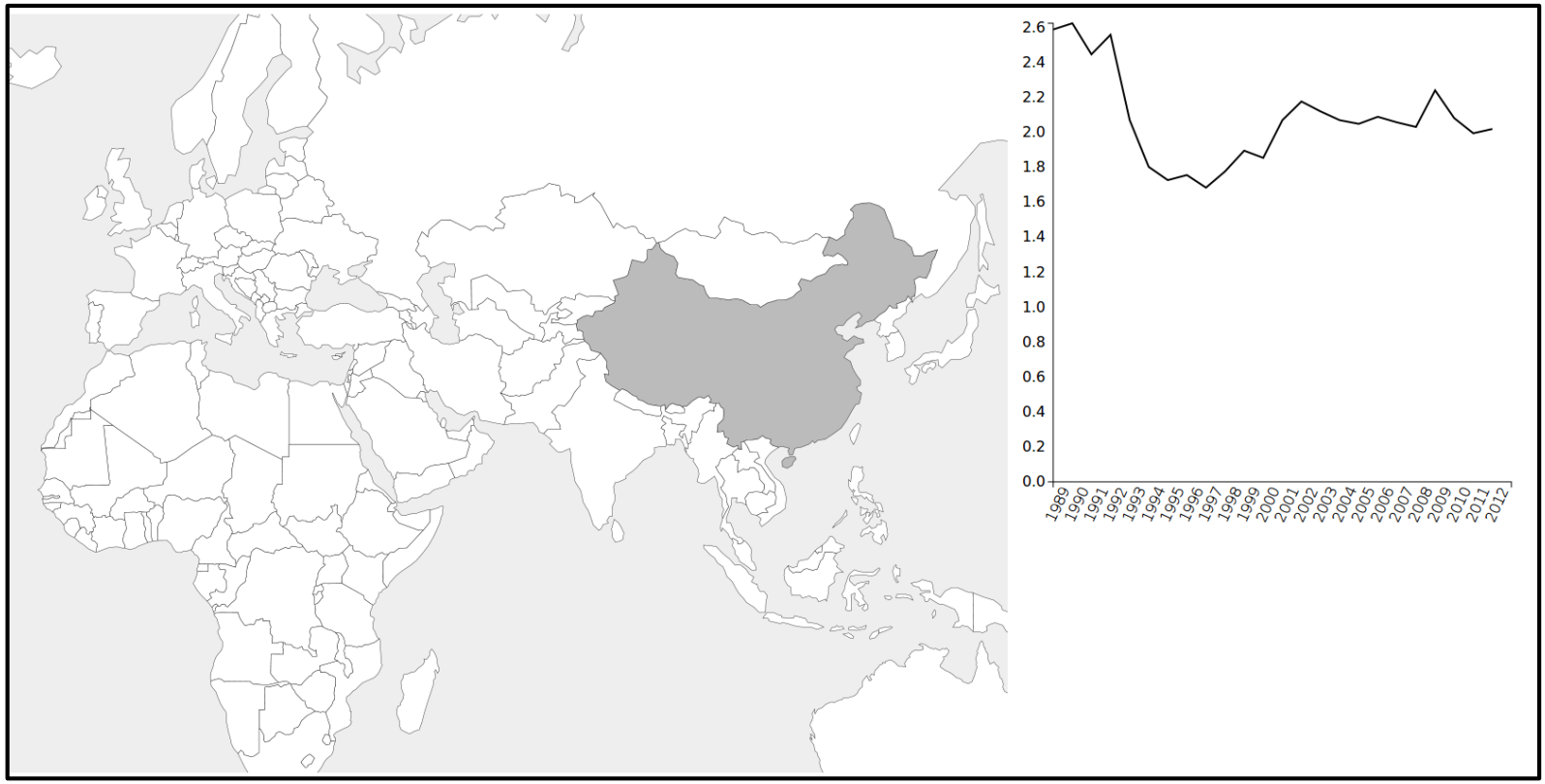
Assignments#3

- _.max([{b: 2}, {c: 3}, {a: 4}, function(v, k){return k;};]) //-> ‘c’
  - minも同じ

- Optionalに関してはAssignments#4のanswersが参考になる

Assignment#4

- multiple_pane.html



Assignments#4

- gdp.csv
  - さっきと一緒

Assignments#4

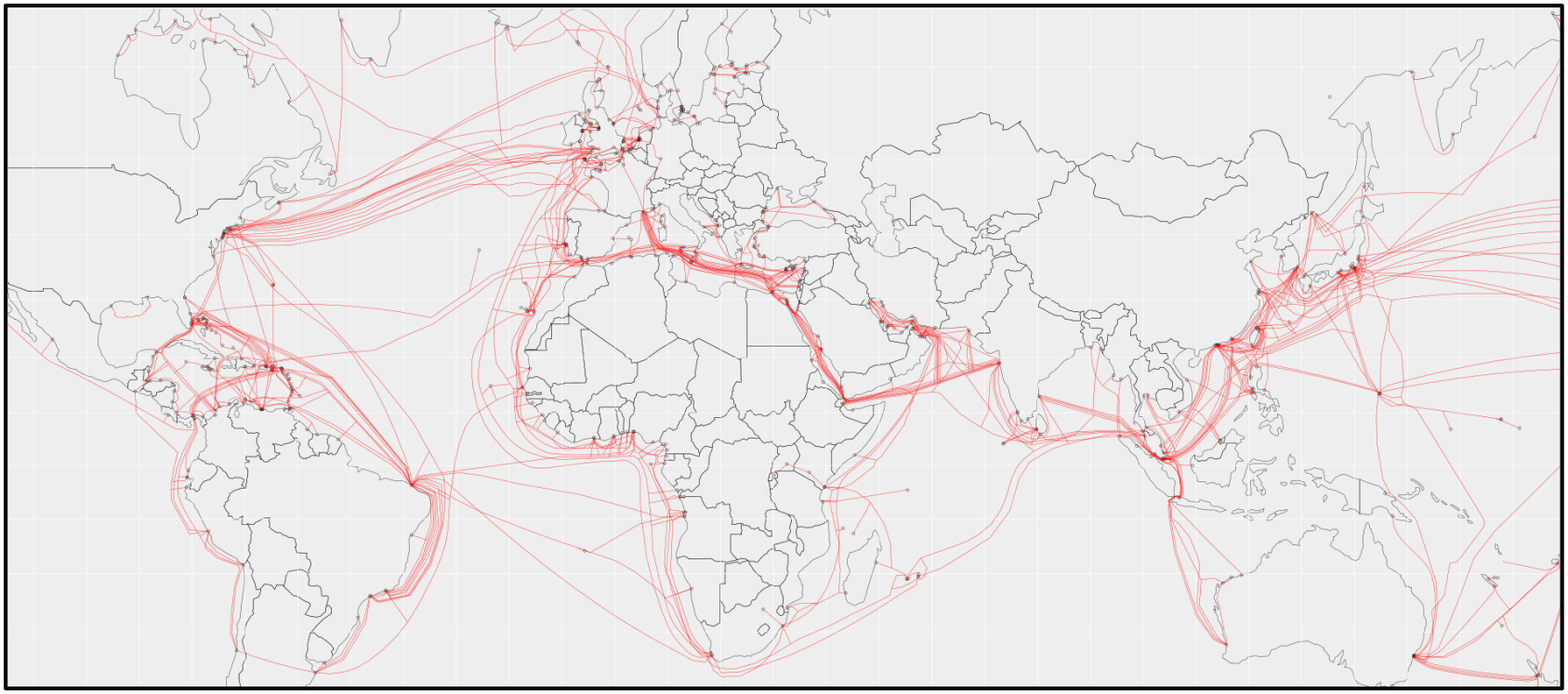
- やること
  - templates/multiple_pane.htmlを改変して左の地図をクリックしたら右に折れ線グラフを表示するようにする(★★☆)
- やること(Optional)
  - 複数の国を選択できるようにする(★★☆)
  - 国によって折れ線グラフ, 地図の色を変えるようにする(★☆☆)
  - 凡例を折れ線グラフの下に表示(★★☆)

Assignments#4

- d3.svg.lineを使う。Assinments#1のスライド参照。
- 国によって色を分けるにはd3.scale.category20()等を使えばよい。
- Optionalの凡例はsvgで描くよりもdivを引っ付けた方がやりやすいかもしれない。

Assignment#5

- cable.html



Assignment#5

- gregs_cable_map.geo.json
  - 全世界の海底ケーブルのデータ
  - ケーブルの座標と接続点の座標
  - データ元:
  - GPL v3

Assignment#5

- やること
  - templates/cable.htmlを改造して海底ケーブルを地図上に表示する(☆☆☆)
- やること(Optional)
  - ケーブルによって色を変える(★☆☆)
  - 地名を地図上に表示する(★☆☆)
  - 各ポイントにマウスを乗せるとマウスの位置に地名が出るようにする等(★★★)

Assignment#5

- データはgeojson形式。
  - 地図データ(https://raw.githubusercontent.com/datasets/geo-boundaries-world-110m/master/countries.geojson)と同じ形式なので地形をpathにしている部分と同じことをする。
- tool-tipを表示するときはtransform属性に気を付けること。