ななちゃんのIT教室

ライブラリに挑戦の巻

by nara.yasuhiro@gmail.com

ななちゃんが
ライブラリに挑戦するという お話

第 0.1 版　2018 年 3 月 27 日

フリー素材
http://freeillustration.net

もくじ

第1回　ライブラリとは

なな：ライブラリって何？　図書館？

先生：今、ここで話題にしているのは、Javascript で便利に使えるプログラム（関数群）を集めたもののことです。

なな：図書館で本を借りて読むように、必要な関数を借りて使うの？

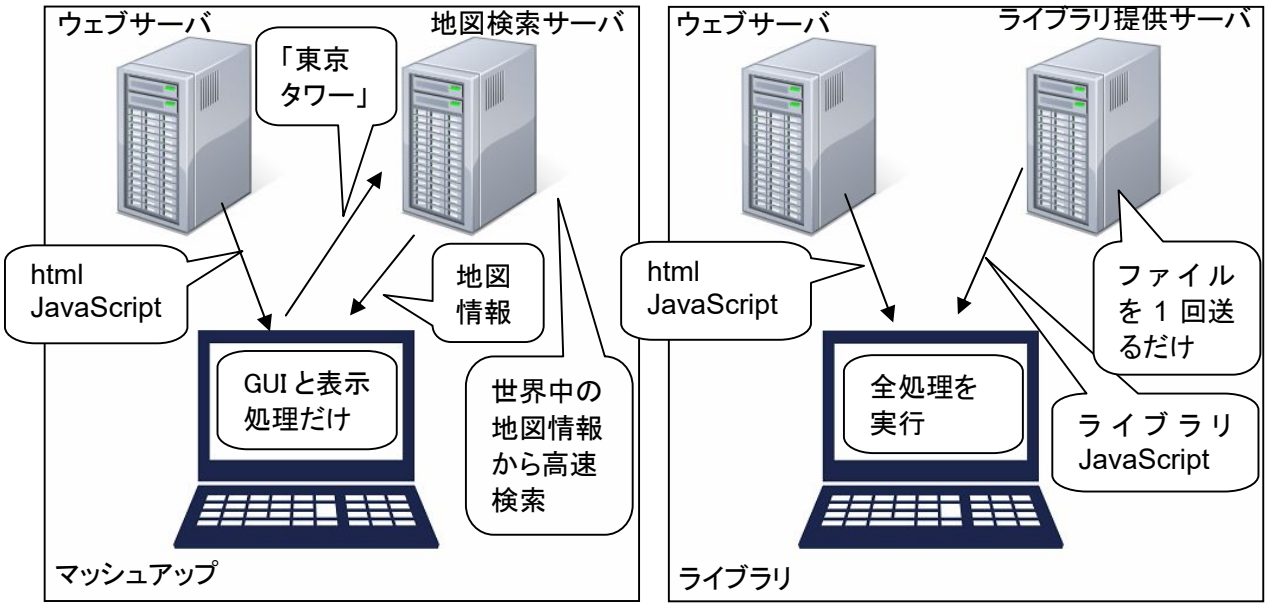
先生：まあ、そんなところね。無料で使えるものがたくさんあります。ライブラリごとに、「使用許諾条件」を確認する必要がありますが、「商用のウェブサイトでも無料で利用して良いもの」が色々あります。

なな：どのように使うの？

先生：<script src="http://提供元サイト/library.js></script> みたいな形で、HTMLファイルで読み込んだり、前もってライブラリのファイルをダウンロードしておき、HTML/JavaScript ファイルと同じ場所に置いて、<script src=library.js></script> みたいな形で読み込んで使います。「library.js」の部分は、ライブラリによって、名前が変わります。実行速度を改善するために、圧縮した library.min.js のような名前で提供されているものもあります。これは、圧縮用のプログラム（ミニファイヤ）で処理したものです。改行を取り除いたり、ユーザが直接呼び出すことのない内部関数の名前を短くしたりして速度を改善する代わりに、人間がソースを見ても理解は困難になっています。

なな：マッシュアップと似ているみたいだけど、違いは？

先生：マッシュアップでは、他者のサーバで動いているプログラム（JavaScriptではないことも多々ある）と通信して、処理を行ってもらいます。どんな形式でデータを送ると、どんな形式で結果を受け取れるかの仕様を API（Application Program Interface）といいます。ライブラリでは、プログラム（普通はJavaScript）を、最初に1回だけダウンロードして、自分のブラウザ上で必要な機能だけ呼び出して実行します。JavaScript のプログラムから関数呼び出しの形で利用します。どんな名前の関数を、どんな引数をつけて呼び出し、どんな値が返されるかの仕様も API といいます。



なな：違いが今いち分かりづらいというか？

先生：たとえば、世界地図の必要な部分を表示する場合、マッシュアップだったら、世界地図すべてのデータを他者巨大サーバに保持して、好きな部分だけを送ってもらうことができます。ライブライだと、巨大な世界全体の地図をダウンロードしたり、ブラウザのところ（クライアントPC上）に保持するのは現実的ではありません。データ更新のたびにダウンロードするのも大変ね。

なな: じゃあ、マッシュアップのほうが便利ということ？

先生: 計算パワーが必要なひとまとめの処理とか、天気予報、為替レートのように時々刻々変化する情報を得るにはマッシュアップが向いているわね。でも、文字列を大文字に変えるとか、郵便番号情報から「ー」を削除するような、細かくて処理量の少ない処理はライブラリのほうが向いているわね。

なな: 欠点というか、気を付けないといけないことは無いの？

先生: マッシュアップの場合、サービスが改版されて、API が少し変わったりすると、自分側のプログラムが動かなくなってしまう心配があります。特に、無償提供のサービスでは、API が変わるだけでなく、サービス提供が終了してしまう可能性もあります。ライブラリは、比較的小さなファイルなので、最新版が発表されても、旧版も提供され続けることが一般的なので、そのような心配がほとんどありません。特に、ファイルを前もってダウンロードしておく使い方をすればプログラムが動かなくなる心配はありません。旧版を使い続けるということで。

なな: ここでも、一長一短なのね。

先生: ライブラリにはこれとは別の、大切な特徴もあります。多くのライブラリは、「いろいろなブラウザに対応している」ということです。自力だけで一生懸命 JavaScript プログラムを作っても、特定のブラウザの古いバージョンでは動作しないというようなことがよくあります。自分のために作って、自分で使うのなら問題ありませんが、みんなに使ってもらうウェブサイトを作る場合には問題になります。変な話みたいですが、自力で作れるような機能でも、あえてライブラリを使うと、幅広いブラウザで利用できるようになるということ。

なな: どうしてそういうことになるの？ ライブラリだと自動的にそうなるの？

先生: 自動的にそうなるんじゃなくて、多くのライブラリ開発者が多くの利用者に使ってもらえるように、日々努力してくれているということなの。利用者が多ければ、利用者からの「あのブラウザで動かないよ」というような意見がたくさん届いて、ライブラリ開発者ががんばってくれるということなの。

なな: 他に注意点は？

先生: フリーソフト全般に共通することだけど、発行部数の多い雑誌などで紹介されているもの、つまり、多くのユーザがある程度以上の期間使っているものを選ぶと安全ね。くれぐれも、小さな意見交換サイトなんかで、「こんな機能のライブラリが欲しい」、「私が作ったのがあるので、よろしければどうぞお使いください」というような形で手に入れたものを使わないこと。重大な欠陥があったり、悪意のあるウィルスとか、個人情報を盗む機能が作りこまれている危険があります。

なな: りょうかい！

第2回　一般的なライブラリ

なな：ライブラリにも、いろいろなのがあるの？

先生：はい。ブラウザの部品、正確には DOM を操作するので有名なのが jQuery。配列やオブジェクト形式のデータを操作すので有名なのが underscore.js と、その改良版の Lodash.js。棒グラフや円グラフなどを扱えるものに、D3.js、chart.js があります。D3.js は、高機能で、多彩なカスタマイズ（機能選択）が可能です。でも、そのぶん、利用するのにプログラミングスキルが必要です。chart.js はカスタマイズはあまりできませんが、すぐに、簡単に使えます。あと、サウンド機能を容易に使える、Tone.js というのもあります。

なな：いろいろあるのね。

先生：細かいのも数えると、数万種類もあると言われているわ。次回から、「ちょっと使ってみる」というのに向いているようなライブラリをいくつか紹介しましょう。

第3回　jQuery

なな：「jQuery」って何？

先生：2006年に開発、公開された JavaScript ライブラリよ。ライブラリは便利な関数群のこと。jQuery は、CSSに近い記述方法を採用しているのが特徴なの。

なな：どんな使い方をするの？

先生：jQuery ライブラリを jQuery のサイト（http://jquery.com/） からダウンロードします。たとえば jquery-3.2.1.min.js というファイル。そして、それを html/JavaScript ファイルの script 要素を使って読み込むの。<script src="jquery-3.2.1.min.js"></script> みたいな形で。また、直接読み込む方法もあります。<script src="http://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script> みたいに。

なな：これは準備作業よね。JavaScript プログラムはどんなふうに書くの？。

先生：
```
<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div>
<script> $("div").css("color", "red"); </script>
```

なな：変わった形をしているのね。

先生：html のどの部分を操作するかを指定（**セレクタ部**）し、その上で操作する内容（**操作部**）を書くのが基本よ。

なな：**セレクタ部**？

先生：「$(...)」 の形をしています。「$」は、「jQuery」と書いても良いんだけど、頻繁に使うので短いほうを使います。具体的には、「（$("div")」 みたいになります。指定した要素、ここでは div 要素の全てが対象になります。JavaScript 的には jQuery （ $ ）という関数が、引数で指定したタグ要素の情報を集め、「jQuery オブジェクト」として返していることになります。

なな：**操作部**は？

先生：例えば、css("color", "red")」。色を赤に指定しています。JavaScript 的には、「jQuery オブジェクト」の css メソッドということになります。

なな： 1 2 3 4 5
<script> $("div").css("color", "red"); </script>
は、どういう動作をするの？

先生：「1」～「5」をすべて赤い文字に変えます。

なな：全体をまとめると、

```
<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div>
<script src="jquery-3.2.1.min.js"></script>
<script>
  $("div").css("color", "red");
</script>
```

こんな感じになって、実行結果は

1
2
3
4
5

こんな感じになるのね。

第4回 Chart.js

数行のコードで折れ線グラフや棒グラフ、円グラフやレーダーチャートなどをダイナミックに表示することができる JavaScript ライブラリ。

グラフを作成できる JavaScript ライブラリとしては D3.js が有名。D3.js は複雑なグラフを描画できるが、グラフの描画処理を自分でコーディングしなければならないので、JavaScript 初級者には使うのが難しい。

Chart.js は D3.js のような複雑な表示をすることはできないが、グラフやチャートの作成に特化しているので、初心者も簡単に扱える。

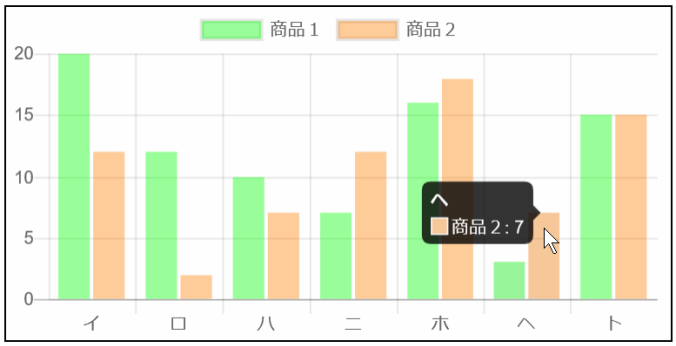
Chart.js は様々な種類のチャート描画に対応している。

- 折れ線グラフ
- 棒グラフ
- レーダーチャート
- 極性面グラフ (鶏頭図)
- 円グラフ・ドーナツ型チャート
- バブルチャート

グラフを作るだけなら、Microsoft Excel でもできるが、Char.js を使えば、データを自動更新して、ウェブ公開できるというメリットがある。

使用例:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <script src="Chart.min.js"></script>
    </head>
    <body>
            <canvas id="canvas" width=450 style="border:solid 1px">
            </canvas>
        <script>
var canvas = document.getElementById('canvas');
var ctx = canvas.getContext('2d');
var myChart = new Chart(ctx, {
   options: { responsive: false },
   type: 'bar',
   data: {
      labels: ['イ', 'ロ', 'ハ', 'ニ', 'ホ', 'ヘ', 'ト'],
      datasets: [{
         label: '商品1',
         data: [20, 12, 10, 7, 16, 3, 15],
         backgroundColor: "rgba(0,255,0,0.4)"
      }, {
         label: '商品2',
         data: [12, 2, 7, 12, 18, 7, 15],
         backgroundColor: "rgba(255,127,0,0.4)"
      }]
   }
});
        </script>
    </body>
</html>
```



第5回　D3.js

D3 (D3.js) は JavaScript ライブラリのひとつ。無料で利用できる。D3 は Data-Driven Documents の略。データの見える化（可視化）を、ウェブ上の標準機能を利用する形で実現している。開発者によれば、このライブラリは明瞭かつ効果的に情報とコミュニケーションできるようにするのが目的であり、データ可視化はその手段にすぎないとのこと。データ可視化のためのシステムは他にもあるが、D3.js の特徴はインタラクション。マウス、キーボードなどの操作を通じて、双方向にデータをやり取りすることができる。

D3 の初版が公開されたのは 2011 年。2016 年に大規模改版が行われ、バージョン 4 となっている。機能は大幅に改善されたが、バージョン 3 との互換性が無いので、解説記事などを調べる場合は注意が必要。非互換部分は多くはないが、古いサンプルプログラムは動作しないものが多い。たとえば、バージョン 4 と 3 のライブラリを両方用意しておき、サンプルプログラムがバージョン 4 環境で動作しない場合、バージョン 3 環境下で動作することを確認してから、非互換部分を修正するなどの工夫が考えられる。

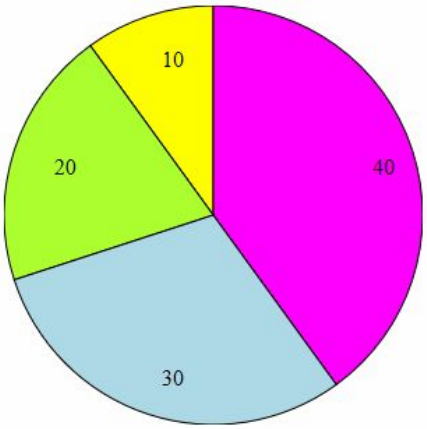
公式サイトは http://d3js.org/、日本語版サイトは http://ja.d3js.node.ws/。

ていて動作効率の良い d3.min.js ファイル。d3.js は動作は同じだが、ソースを読んで研究する場合に適している。

お勧めの参考書。ただし、d3.js version 3 ベース

円グラフ

円グラフは英語で pie chart という。




```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <style>svg { width:300px; height:300px }</style>
        <script src="d3/d3.min.js"></script>
    </head>
    <body>
        <svg id="chart"></svg>
        <script>
var width = 300;
var height = 300;
var data = [ 10, 40, 20, 30 ];

var pie = d3.pie().value(function(d){ return d; });

var g = d3.select("#chart")
      .selectAll("g.arc")
      .data(pie(data))
      .enter()
      .append("g")
      .attr("class", "arc")
      .attr("transform", "translate(" + (width / 2) + "," + (height / 2) + ")");

var arc = d3.arc().innerRadius(0).outerRadius(width / 2);
var color = d3.scaleOrdinal()
      .range(["yellow", "magenta","greenyellow", "lightblue"]);

g.append("path")
      .attr("d", arc)
      .style("fill", function(d) { return color(d.data); })
      .attr("stroke", "black");

var labelArc = d3.arc()
      .outerRadius(width/2 - 30)
      .innerRadius(width/2 - 30);

g.append("text")
        .attr("transform", function(d) { return "translate(" + labelArc.centroid(d) + ")"; })
        .attr("dy", "0.5em")
        .text(function(d) { return d.data; });
        </script>
    </body>
</html>
```

<version 3>　<version 4>
d3.layout.pie()　-> d3.pie()
d3.svg.arc()　-> d3.arc()

pie() は、データを降順にならべかえ、 データごとに扇形（パイ形）の開始角、終了角情報を含むオブジェクトを生成するメソッド

g はグループタグ

表示位置を調整する

<version 3>
d3.scale.ordinal()

扇の形を生成

var data = [10, 40, 20, 30];に対する色

扇と扇の間に黒線を入れる

扇にテキストを挿入

扇の中に書き込むテキストの位置

第6回　Tone.js

簡単に「Web Audio API」を活用できる JavaScript ライブラリ。https://tonejs.github.io/。

「Tone.js」ライブラリは、GitHub（https://github.com/Tonejs/Tone.js）からダウンロードするか、以下の URL から利用可能。https://tonejs.github.io/build/Tone.min.js。例：<script src="Tone.min.js"></script>。

```
var synth = new Tone.Synth().toMaster();
synth.triggerAttackRelease('C5', '2n');
```

「Tone.Synth()」
　音源に接続（この場合は「Synth」という音源）。その音源を「マスター」と呼ばれる場所「toMaster()」に接続。
　これを変数「synth」に格納し、「synth.triggerAttackRelease()」で音を鳴らす。「C5」：第 5 オクターブの「C」（ハ長調のド）。「”2n”」：2 分音符、「”4n”」：4 分音符、「”8n”」：8 分音符、「”1m”」：全音符(1 小節)、相対"+4n"。

「triggerAttackRelease()」を複数行書くことで、「単音」だけでなく、「音階」（単音列）を鳴らすことができるが、音が同時にならないように、時刻（何秒後か）を指定する。

```
<!doctype html>
<html>
      <head>
            <meta charset="utf-8">
      </head>
      <body>
          <script src="Tone.min.js"></script>
          <script>
var synth = new Tone.Synth().toMaster();
synth.triggerAttackRelease('C5', '2n',1);
synth.triggerAttackRelease('D5', '2n',3);
synth.triggerAttackRelease('E5', '2n',5);
          </script>
    </body>
</html>
```

「Tone.js」には「シーケンス制御」が可能な API も提供されている。

```
var melody = new Tone.Sequence(【関数】, 【音階の配列】).start()
```

```
var melodyList = [ 'C3', 'D3', 'E3', 'F3', 'G3', 'A3', 'B3', 'C4' ];

function setPlay(time, note) {
      synth.triggerAttackRelease(note, '8n', time);
}

var melody = new Tone.Sequence(setPlay, melodyList).start();

Tone.Transport.start();
```

関数の引数「time」に何秒後に音を出すかという情報が入れられた上で呼び出される。音符ごとに時刻を割り振ることで「シーケンス」で制御された「音階」が再生される。

音階は、周波数を表す数字（例：440）、音階＋オクターブを表す文字列（例：”F#6”、"Bb4"）。

```
<!doctype html>
<html>
	<head>
		<meta charset="utf-8">
	</head>
	<body>
		<script src="Tone.min.js"></script>
		<script>
var melodyList = ['C3', 'D3', 'E3', 'F3', 'G3', 'A3', 'B3', 'C4'];
var synth = new Tone.Synth().toMaster();

function setPlay(time, note) {
	synth.triggerAttackRelease(note, '8n', time);
}

var melody = new Tone.Sequence(setPlay, melodyList);
melody.start();
melody.loop = 2;
Tone.Transport.start();
		</script>
	</body>
</html>
```