SVG 資料第 5 回目 (その 3)

JavaScript の配列とオブジェクトリテラル、要素を作成する関数群

メディア専門ユニット I(SVG)

2017/5/23

JavaScript の配列

JavaScript の配列は次のような特徴を持つ。

- 配列の要素のデータ型はすべて同じでなくてもよい
- 配列に要素を追加したり削除したりできる
- 配列の大きさは `length` プロパティで得られる

「開発者ツール」のコンソールで確かめてみる

配列の操作

```
>A = [];          //変数 A を空の配列で初期化
[]
>A.length         //配列の大きさは中身がないので 0
0
>A[5]= 1;         //添え字が 5 の要素を 1 に設定
1
>A.length         //A の長さは 6 になる
6
>A[0]             //A[0] から A[4] までは値が代入されていないので
undefined         //undefined(定義されていない) となる
```

A=[1,2,"Test"] として配列を操作してみよう

オブジェクトリテラル

- オブジェクトリテラルとはキーと値のペアのリストのこと
- 全体を波かっこ ({}) で囲む
- ペアと値の間はコロン (:) で区切る
- 各ペアはコンマ (,) で区切る

オブジェクトリテラル

```
>A={red:"赤","blue":"青","dark-green":"#004000"};
Object {red: "赤", blue: "青", dark-green: "#004000"}
  //3 つのペアが定義されている
  //キーは\texttt{"}ではさまなくてもよい
 //最後のキーのように変数名として使えない文字を含むものは
 //\texttt{"}ではさむ
>A.red;//値は変数のメンバーのようにアクセスできる
"赤"
>A.blue;//文字列であっても同じ
"青"
>A["red"];//文字列を添え字のように使える
"赤"
>A["dark-green"]//-が演算子と重なるのでこの方法でしかアクセスできない
"#004000"
>for(let name in A) console.log(`A["${name}"]=${A[name]}`);
  //配列のすべての要素を渡るために for( .. in ..) の構文を用いる
A["red"]=赤
A["blue"]=青
A["dark-green"]=#004000
```

要素を操作、作成するための関数群 (1)

```
 1function MKHTMLElm(P, Elm, Attribs, Events) {
 2  let HTMLNS = "http://www.w3.org/1999/xhtml";
 3  return MakeElement(HTMLNS, P, Elm, Attribs, Events)
 4}
 5function MKSVGElm(P, Elm, Attribs, Events) {
 6  let SVGNS = "http://www.w3.org/2000/svg";
 7  return MakeElement(SVGNS, P, Elm, Attribs, Events)
 8}
 9function MakeElement(NS, P, elem, attribs, events) {
10  let Element = document.createElementNS(NS,elem);
11  SetAttributes(Element, attribs);
12  AddEvents(Element, events);
13  if(P) P.appendChild(Element);
14  return Element;
15}
```

要素を操作、作成するための関数群 (1)
コードの解説

- ここでは HTML 内の要素と SVG 内の要素を作成する関数を定義
  - HTML 内の要素を定義する関数は `MKHTMLElm()`(1 行目から 4 行目)
  - SVG 内の要素を定義する関数は `MKSVGElm()`(5 行目から 8 行目)
  - 引数は新規作成要素の親要素、要素名、属性のリスト、イベント処理のリスト
  - それぞれの名前空間を引数に追加して `MakeElement()` を呼び出し、作成した要素を戻り値とする
- 9 行目から 15 行目で `MakeElement()` が定義
  - 10 行目で与えられた名前空間の要素を作成
  - 11 行目で作成した要素の属性を設定
  - 12 行目で作成した要素にイベント処理を設定
  - 13 行目で指定された親要素が `null` でなければ作成した要素をその子要素に設定

要素を操作、作成するための関数群 (2)


```
16 function SetAttributes(Elm, attribs) {
17   for( attrib in attribs) {
18       Elm.setAttribute(attrib,attribs[attrib]);
19   }
20 }
21 function AddEvents(Elm, Events) {
22   for( event in Events) {
23       Elm.addEventListener(event,Events[event][0], Events[event][1]);
24   }
25 }
26 function RemoveEvents(Elm, Events) {
27   for( event in Events) {
28       Elm.removeEventListener(event,Events[event][0], Events[event][1]);
29   }
30 }
```

要素を操作、作成するための関数群 (2)
コードの解説

- 16 行目から 20 行目でオブジェクトリテラルの形式で与えられた属性と属性値のペアに対いして `for in` ループで設定
- 21 行目から 25 行目でオブジェクトリテラルの形式で与えられたイベント処理のリストに基づき指定された要素にイベント処理を設定
- 26 行目から 29 行目ではオブジェクトリテラルの形式で与えられた取り除くイベント処理のリストに基づき指定された要素からイベント処理を取り除く

これらの関数群を使用して「ドラッグして直線を引く」を書き直してみよう

関数群を利用した例

乱数を使用して円をいくつか描く

```
1<?xml version="1.0" encoding="UTF-8" ?>
2<svg xmlns="http://www.w3.org/2000/svg"
3     xmlns:xlink="http://www.w3.org/1999/xlink"
4     width="100%" height="100%">
5  <title>ランダムな色と大きさを一定の時間で表示し続ける</title>
6  <script type="text/ecmascript" xlink:href="make-svg-elm.js"></script>
```

- 6 行目で関数群が記された JavaScript ファイルを読み込む
- ファイル名は属性 `xlink:href` で指定することに注意

乱数を使用して円をいくつか描く ソースコード (2)


```
 7  <script type="text/ecmascript">
 8//<![CDATA[
 9let Objs=[], Canvas, Unit = 15, xMin = 40, xMax = 400, yMin = 40, yMax = 300;
10let Colors = ["red", "yellow", "blue", "pink"];
11window.onload = function() {
12  let id, r;
13  Canvas = document.getElementsByTagName("g")[0];
14  for(id = 0; id <20; id++) {
15    r = getRandVal(1, 4);
16    Objs[Objs.length] = MKSVGElm(Canvas,"circle",
17       {"id":id, "cx":getRandVal(xMin, xMax), "cy":getRandVal(yMin, yMax),
18                 "r":Math.sqrt(r)*Unit,
19                 "fill":Colors[getRandVal(0, Colors.length)], "opacity":0.6,
20                 "stroke-width":"2", "stroke":"black"}, {});
21  }
22  window.setTimeout(renew, getRandVal(10,20)*100);
23}
```

乱数を使用して円をいくつか描く ソースコード (2)

```
24function renew() {
25  for(let id = 0; id <20; id++) {
26    SetAttributes(Objs[id],
27      {"cx":getRandVal(xMin, xMax), "cy":getRandVal(yMin, yMax),
28       "r":Math.sqrt(getRandVal(1, 4))*Unit,
29       "fill":Colors[getRandVal(0, Colors.length)]},{});
30  }
31  window.setTimeout(renew,getRandVal(10,20)*100);
32}
33function getRandVal(Min, Max) {
34  return Math.floor(Min+(Max-Min)*Math.random());
35}
36//]]></script>
37<g />
38</svg>
39
40
41
```

乱数を使用して円をいくつか描く ソースコード (2) 解説

- 9 行目の変数で表示する円の座標の範囲を示す 4 つの変数の値を定義
- 12 行目で 25 行目にある<g>要素を得ている
- 14 行目から 19 行目で一つの円を新規に作成し、25 行目の<g>要素の子要素にしている
- 属性 cx、属性 cy、属性 r は乱数を用いて選択
- 属性 fill の色は 10 行目の配列 Colors から乱数で選択
- 乱数を求める計算は 21 行目から 23 行目の getRAnVal()
- 引数は最小値 (Min) と最大値 (Max)
- Math.random() は 0 から 1 の値を返すので、その値を最小値と最大値になるように変換している
- 戻り値を整数とするために Math.floor() を用いる

やってみよう

- 10 行目の配列に色を追加してもプログラムが正しく動くことを確認し、その理由を述べよ
- 円の代わりに (形の異なる)3 角形をランダムに表示するものを作成する