情報メディア専門ユニットI(セミナー) (2018 年度版)

1　全般的な注意

この授業の進め方と注意はつぎのとおりである。

1. セミナー開始時にくじを引いて 3 人のグループを 3 つ作成しその中で議論を行う。
2. グループ内で検討した結果を 1 グループあたり 10 分を目安に最後に発表する。
3. セミナーでは与えられた課題をグループごとに行った議論については詳細なメモをノートにまとめ、その日の発表に生かすこと。
4. ノートは授業終了時に提出するかは今のところ未定である。

この授業で使用するブラウザは Google Chrome とする。他のブラウザでもよいが、ブラウザによっては機能が対応していないこともある。各種ブラウザの表示の違いなどを確認しておくのもよいであろう。

2　第 1 回 — SVG による基本図形の描画

2.1　作業内容とレポート

次の事柄について確認しながら作業し、レポートとして報告すること。今回の内容はテキストの第 2 章を参考にすること。

1. www.hilano.org/hilano-lab/の情報メディア演習 I の資料の中からいくつかの SVG の画像を表示できることを確認する。ブラウザーは Chrome で確認すること。
2. 長方形をいくつが色や形を変えて描く。
3. 円や楕円をいくつが色や形を変えて描く。
4. 内部の塗りをグラディエーションにする。
5. テキストにある錯視図形の問題をいくつか作成すること。また、それらの一部の属性を変えて見え方がどのように変化するか報告すること。

レポート課題 1 次の事柄について報告すること。

1. 各種ブラウザーで表示が異なる点があれば報告すること。
2. 本日の演習で利用した要素とその属性についてまとめる。
3. BOM とはなにか。SVG ファイルで BOM があるとなぜいけないのか。
4. 図形が描かれる順序と表示の形につて説明する。
5. 作成した図形のついて解説をつける。

レポート課題 2 (余力問題) スマートフォンなどの携帯端末のブラウザで SVG の画像が表示できるか調査すること。

3　第 2 回 — SVG におけるアニメーション

3.1　アニメーションに関する注意

アニメーションに関する基本的な事項を確認する。

- アニメーションの種類
  - `animateTransform` 位置を移動する
  - `animateColor` 色を変化させる
  - `animateMotion` 道のりに沿ったアニメーション
    道のりについては次回行うので今回は範囲外とする。
  - `animate` 上記以外の属性を変化させる
  - `set` 値を瞬時に変化させる
- アニメーションの要素はその親要素につくことを忘れないように

  ```
  <rect x="0" y="0" width="100" height="200" fill="red"/>
  ```

  の長方形の `fill` にアニメーションをつけるには`<rect` の最後にある`/>`を`>`に変える必要がある。

  ```
  <rect x="0" y="0" width="100" height="200" fill="red">
    <aniamteColor attributeName="fill" attributeType="CSS"
      from="red" to="green" dur="10s" fill="freeze"/>
  </rect>
  ```

- いくつかの画像をまとめて移動させたいときは親要素として`<g>`を追加するとよい。
- 不透明度 (配布資料 35 ページ) についても理解すること

3.2　課題

次の事柄からいくつか選択して作業すること。

1. 情報メディア演習 (セミナー) の資料からいくつかを選び、各種アニメーションが実行できることを確認する。

   アニメーションのキャプチャ画像は開始時（に近いとき)、途中、終了時の 3 つを添えること
2. いろいろな図形が移動するアニメーションを作成する。
3. 色のアニメーションで次のことを行う。
   - アニメーションの色を変える。
   - 複数の図形に別の色のアニメーションをつける
   - 図形がある範囲で左右に動き、動きの向きが変わるときに一瞬、色が変わる
   - 不透明度にアニメーションをつけて、重なった図形の見え方を変える図形

4. グラデーションのアニメーションで次のことを行う。

   - アニメーションが左から右に流れる。
   - 属性 `x1` と属性 `x2` のアニメーションのスピードを変える
   - 属性 `stop-color` や属性 `offset` にアニメーションをつける

5. 前回作成した図形に、位置の変化、色の変化のアニメーションをつけたものを作成すること。1 年の時の Proccessing で作成したものを SVG で作り直すのもよい。そのときは作成の手間などの比較もするとよい。

6. テキストにある錯視図形のいくつかにアニメーションをつけ、効果の変化を調べること。たとえば次のようなものが挙げられる。

   - Fick の錯視で垂直線を水平線の左端から右端に移動させる (問題 4.1)。
   - Judd の錯視 (問題 4.3) を作成し、両端の線分の回転させて開く角度を変化させるアニメーションをつける。
   - カフェウォール錯視 (23 ページ) の細い線に色のアニメーションをつける
   - 問題 4.11 の図を作成する。

7. **(ちょっと面倒)** グラデーションにアニメーションをつける図形を`<stop>`要素のアニメーションだけで作成する

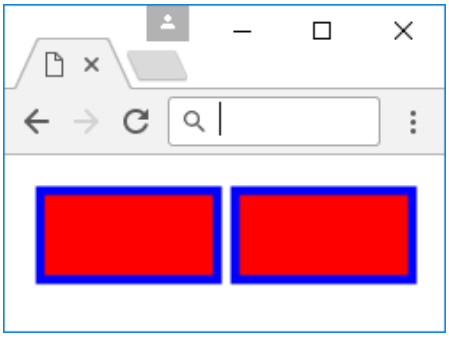
4　第 3 回 — 複雑な図形の描き方、文字列

4.1　課題

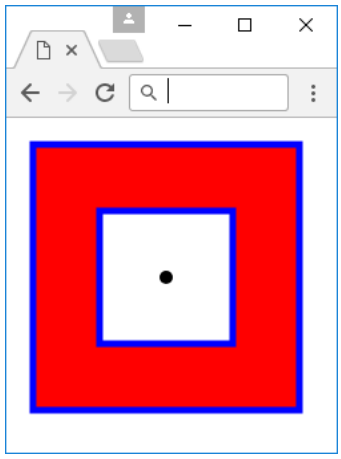
次の中からいくつか選んで報告する。

1. `polyline` または `polygon` を用いた図形を描くこと。たとえば次のようなことを考えてください。

   - Excel など他のプログラムを用いて正 6 角形の頂点の座標を計算し、その結果を用いて正 6 角形を描く
   - 一つの`<path>`要素で二つの長方形を描く

   - 穴が開いた正方形を一つの`<path>`要素で描く

   - `<polygon>`要素、`<polyline>`要素と`<path>`要素での描き方の比較をする
   - `<path>`要素を用いて曲線を含む図形を作成する
   - `path` を用いた図形を描くこと

   なお、これらの図形には必要であればグラデーションやアニメーションをつけることが望ましい。

2. 今まで作成した図形の繰り返しがある画像を選んで（なければ今回新たに作成して）`pattern` を用いて塗りつぶすこと。`pattern` を用いなかった場合と比較をして考察に加えること。

- 今までに作成したSVG画像で繰り返しの部分があるものをパターンで書き直し、コードの長さ、変更の手間などの比較、考察をする。
- ザビニの錯視 (配布資料 46 ページ) や輝くヘルマン格子、モーガンのねじれのひも (配布資料 48 ページ) をパターンを用いて作成する。
- パターンを構成する要素の一部の属性にアニメーションをつける。錯視図形の場合にはその見え方の変化を報告する。

3. 文字列を表示すること

- 自分の名前を表示
- それを姓と名前の部分に分け、移動したり、色を変えたアニメーションをつける
- 文字列表示の属性 (配布資料 101 ページ) の属性 `text-anchor` や属性 `dominant-baseline`（表 5.1 を参照）の違いを例とともに報告する。ブラウザによって対応が異なるかもしれない。

5　第 4 回 — イベント処理

今回からは使い慣れたブラウザで作業をしてかまわない。

5.1　開発者ツールの使い方を理解する

この課題は必ず行うこと。

1. SVG のファイルを開き、「開発者ツール」を開く方法を確認する。
2. SVG ファイルの要素の属性値を直接、変えることができること (Elements)
3. コンソールを開き、そこで簡単な式を打ち込んで計算結果が表示されることを確認
4. 3 つの円の塗りつぶしを表示するリストを打ち込んで実行する。また、表示する属性値を変える。

次のことを確認する。

1. 使用したブラウザとそのバージョン
2. 「開発者ツール」に相当する機能の名称
3. 「開発者ツール」を開くショートカットキー
4. どこかの Web ページを開いて DOM ツリーのノードを展開し、DOM ツリー上でマウスカーソルを動かしたときの現象
5. SVG ファイルを開いて要素のプロパティを修正する
6. コンソールの入力がどこでできるか確認する。
7. コンソールで `2+3;` と入力して結果が表示されることを確認
8. コンソールで次のように入力したときの結果を確かめる (結果の展開が可能なはずなので確認すること)

```
document.getElementsByTagName("circle");
```

なお、JavaScript の実行中のエラーメッセージは Console に表示される。

5.2　イベント処理の基本

リスト 6.3 に追加、変更して次のことを行う。

1. 円に代わりに正方形をいくつか置いてそのうえでクリックしたときに移動する
2. クリックした要素の色を変える (`red`⇒`yellow`⇒`blue`⇒`red` のように変化するようにできるか)
3. 異なる種類の要素を置いてクリックしたときに移動する (`tagName` プロパティを使う)

リスト 6.1 を利用して円を一つ表示した図形に対して次のことができるようにする。

1. 円に対してもイベント処理関数を登録してその上をクリックしたときに移動するようにする
2. 円の上をクリックしたら色が変化する。それ以外のときは移動する。
3. 2 つ以上の要素を置く。要素以外のところでクリックしたら最後にクリックした要素が今回クリックした位置に移動する

5.3 ドラッグの処理

リスト 6.8 で次のことを確認する。

- ドラッグ処理のプログラムを実行して動きを確認する。
- 「開発者ツール」の Elements を開いてドラッグするごとに DOM ツリーが変化すること

さらに次のことができるようにする。

1. 正方形をドラッグするように変える
2. 文字をドラッグするように変える
3. 通常、アイコンなどをドラッグするときと、ここでのドラッグとの相違点を述べ、それを改善する
4. `<path>`要素で描かれた図形をドラッグするものを作成する

5.4 イベント処理に関する質問

次の事柄について説明をする。

1. イベントオブジェクトの `target` と `currentTarget` の違い
2. イベント処理の関数はどの要素に付けるのが良いか。
3. 円と正方形の図形に対し、1 種類のイベント処理関数でする方法を考えよ。図形の種類が増えてもイベント処理関数に手を付けないようにするにはどのような方法があるか検討すること

最後の課題では自分の親要素を示す `parentNode` を使うとよいかもしれない。

6　第 5 回 — イベント処理と要素の作成

6.1　クリックした位置を SVG 内に表示

リスト 6.5 につて次の事項を行う。

- 25 行目と 27 行目の<text>要素の中の空白をなくすと正しく動かないことを確認
- 空白があってもなくても動作するように修正する (配布資料参照のこと)
- 23 行目にある<g>要素を取り除いて、イベント処理関数を 24 行目の<rect>要素につけない理由はなにか
- このリストの不備な点を指摘し、改善をする
- 表示する図形を円から別なものに変えて、その位置を示す属性の値を表示する

6.2　要素の追加

SVG 文書に新たに要素を付け加える方法を学ぶ。
リスト 6.7 について次の事項を行う。

- 何本かの直線を引いた後で次のことを確認
  - 右ボタンでクリックしてソースコードが変化しているか
  - DOM ツリーを見て、作成した直線の要素があるか
- SVG 内に小さな正方形で塗りの色が異なるものをいくつか置き、その正方形をクリックした後では、クリックした正方形の塗りの色で直線が引けるようにする ((配布資料 96 ページ) 図 6.12 参照)
- 長方形をドラッグで描く

6.3　初期化で要素を作成と自前のアニメーション

- 複数の直線からなる図形を JavaScript のプログラムで作成する (正 n 角形など)
- その図形の直線を 1 本ずつ付け加えるアニメーションを作成する
- processing で作成したアニメーションを JavaScript で再現し、手間の比較を検討する
- 円を一定時間だけ表示する (要素を取り除くためには `removeChild` を用いる)

6.4　要素を操作、作成するための関数群の作成

次の課題を行う。

- 「ドラッグして直線を描く」の SVG 文書をリスト 6.12 の関数を利用して書き直し、両者の比較を行う
- 乱数を利用して一定の時間だけ円をいくつか表示する
- 前問に対して表示されている間クリックされた円の数を数える機能を追加する

7　第 6 回 — インライン SVG と HTML 要素の間のデータの交換

7.1　HTML のフォームについて

次のリストを入力して次のことを行う。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <title>HTML の例</title>
    <script type="text/ecmascript">
      //<![CDATA[
      window.onload = function(){
        let F = document.getElementsByTagName("form")[0];
        F.addEventListener("change",change,false);
        F.addEventListener("click",click,false);
      }
      function change(E) {
        console.log(`change::target:${E.target.tagName}`+
                    `, E.currentTarget:${E.currentTarget}`+
                    `, value:${E.target.value}`);
      }
      function click(E) {
        console.log(`click::target:${E.target.tagName}`+
                    `, E.currentTarget:${E.currentTarget}`+
                    `, value:${E.target.value}`);
      }
      //]]>
    </script>
    <style>
      .head {
      background:rgb(128,128,255);
      width:180px;
      text-align:center;
      font-size:18px;
      }
    </style>
  </head>
  <body>
    <h3>フォームのサンプル</h3>
    <form>
      <div>テキストボックス<input type="text" size="20"></input></div>
      <div>パスワード<input type="password" size="20"></input></div>
```

```
      <div class="head">チェックボックス</div>
      <div><input type="checkbox" value="check1">チェック 1</input></div>
      <div><input type="checkbox" value="check2">チェック 2</input></div>
      <div class="head">ラジオボタン 1</div>
      <div><input type="radio" name="R1" value="radio1-1">ラジオボタン 1-1</input></div>
      <div><input type="radio" name="R1" value="radio1-2">ラジオボタン 1-2</input></div>
      <div><input type="radio" name="R1" value="radio1-3">ラジオボタン 1-3</input></div>
      <div class="head">ラジオボタン 2</div>
      <div><input type="radio" name="R2" value="radio2-1">ラジオボタン 2-1</input></div>
      <div><input type="radio" name="R2" value="radio2-2">ラジオボタン 2-2</input></div>
      <div><input type="radio" name="R2" value="radio2-3">ラジオボタン 2-3</input></div>
      <div class="head">ボタン 2</div>
      <div><input type="button" value="押してね"></input></div>
      <div class="head">プルダウンメニュー</div>
      <div>
        <select>
          <option value="option1">オプション 1</option>
          <option value="option2">オプション 2</option>
          <option value="option3">オプション 3</option>
        </select>
      </div>
    </form>
  </body>
</html>
```

- フォームにおけるそれぞれのデータの入力の取り扱いについてまとめる。
- プルダウンメニュー、ラジオボタンの要素を追加や変更を行う
- ラジオボタンのどこかをチェックした後でコンソールで次の入力を行った結果を報告する

  `document.querySelector("input[name=\"R1\"]:checked")`

  また、次のようにしたらどうなるか確認する。

  `document.querySelector("input:checked")`
- 上のことを踏まえてフォームにどのようなデータが入っているかを知る方法について、要素ごとにまとめる

7.1.1 HTML のフォームのデータと SVG の間でのデータ交換

リスト 6.16 において次のことを行う。

1. 表題のフォントを変えて (CSS 内の `.display` の内容を変える) もクリックした位置が変わらないことを確認する。
2. 実行の途中でブラウザの横幅を変えて表題の行数を変化させるとクリックした位置と円の移動位置が変わることを確認する。

3. その状態で再読み込みをするとマウスのクリック位置に円が移動することを確認する。

4. この不具合を直すこと。

5. 円の色の選択を増やすことをしなさい。

6. 円以外の形に変えて、それらの属性を HTML の方から設定できるようにすること