# ソフトウェア開発
# 第 13 回目授業 (補講)

平野 照比古

2017/12/27

# 例の概要

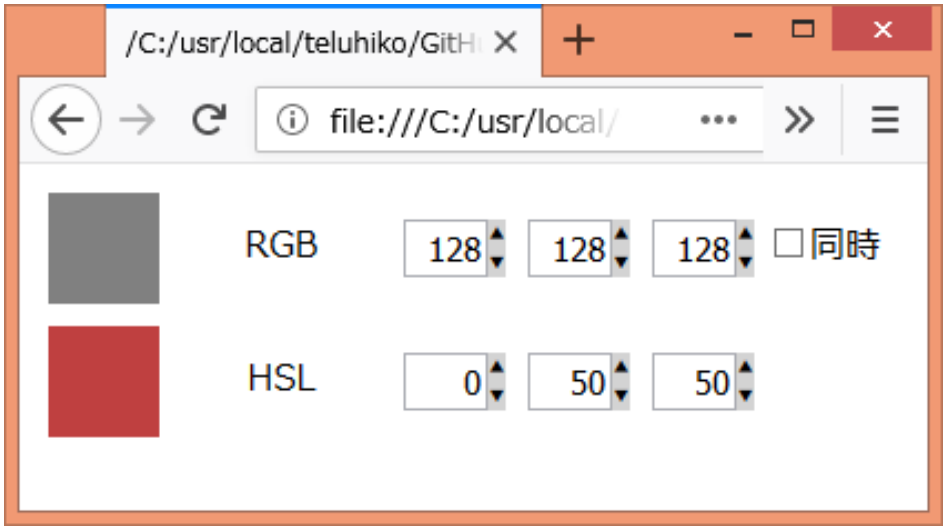


Figure: 色の指定を見る

- 上下 2 行のテキストボックスで指定された色を左側の正方形の部分で表示
- 上は RGB 形式で、下は HSL[1] 形式で色を指定

[1] CSS Color Module Level 3
https://www.w3.org/TR/2011/REC-css3-color-20110607/を参照

# 色を変える操作

- れぞれのテキストボックスの値は▲をクリックすると増加し、▼をクリックすると 1 ずつ減少
- シフトキーを押しながらクリックすると 5 ずつ変化
- RGB 形式の値は $0 \sim 255$ の間で変化する。上限または下限の範囲を超える場合は上限または下限の値にのままで一番右のチェックボックスをチェックすると 3 つの値が同時に増減
- HSL 形式の値は一番左 (H–色相) が $0 \sim 359$ の間で循環して変化し、残りの 2 つは $0 \sim 100$ の間で変化

# HTML ファイルのソース

```
1 <!DOCTYPE html>
2 <html>
3   <meta charset="UTF-8"/>
4   <head>
5     <script type="text/ecmascript" src="13Ex.js"></script>
6     <script type="text/ecmascript" src="13UI.js"></script>
7     <style type="text/css">
8       .color{
9       width:50px;
10      height:50px;
11      display:inline-block;
12      vertical-align:middle;
13      margin:5px;
14      }
15      </style>
16  </head>
17  <body>
18    <div id="RGB"><div class="color"></div></div>
19    <div id="HSL"><div class="color"></div></div>
20  </body>
```

# HTML ファイルのソース–解説

- 5 行目ではこのページに関する処理を定義する JavaScript ファイル (`13Ex.js`) を読み込む。
- 6 行目ではユーザーインターフェイスを定義している JavaScript ファイル (`13UI.js`) を読み込む。
- 7 行目から 15 行目は色を表示する部分の CSS を定義している。
  - 9 行目と 10 行目では表示部分の大きさ
  - 11 行目では配置方法 (横に並べる)
  - 12 行目では上下の位置 (ここでは中央に指定)
  - 13 行目では色の表示域の周りの空白
- 18 行目と 19 行目では色の表示位置と値を設定するための`<div>` 要素を定義

# オブジェクトのオプションの値を変更する関数

```
1 function setOptions(props, Opt) {
2   for( let key in Opt) {
3     if(props.hasOwnProperty(key)) props[key] = Opt[key];
4   }
5 }
```

- 2 番目の引数 (Opt) は変更する値が入っているオブジェクト
- 1 番目の引数はキーがオブジェクト内の指定できるオプション (デフォルト値が設定されている) からなるオブジェクト
- 3 行目で指定できるキーであれば値を設定

# TML 要素の style を設定する関数

```
 6 function setStyle(props, Opt) {
 7   setOptions(props, Opt);
 8   return {
 9     style: JSON.stringify(props).replace(/[{"]/g,"").replace(/[,}]/g,";")};
10 }
```

- 7 行目で、指定されたオプションを設定
- 9 行目でオプションのオブジェことを style 属性の形式になるように変更
  1. オブジェクトを JSON 形式の文字列に変換
  2. キーなどを囲む { と " を取り除く
  3. , と } を ; に変換

# 基本となるオブジェクト DOMObject(1)

```
11 let DOMObject = (()=>{
12   let NS ={
13     HTML:"http://www.w3.org/1999/xhtml",
14     SVG: "http://www.w3.org/2000/svg"
15   }
```

- オブジェクト内で有効な定数を定義するためにクラス式を返す関数を定義しその場で実行している (43 行目の ())。
- 12 行目から 15 行目で作成する要素の名前空間のリストをオブジェクトリテラルの形で定義している。ここでは HTML 要素と SVG 要素の名前空間がある。

# 基本となるオブジェクト DOMObject(2)

```
16   return class{
17     static getElmId(id) {
18       return {elm:document.getElementById(id)};
19     }
20     static getElmsTagName(elm) {
21       let elms = document.getElementsByTagName(elm);
22       return Array.prototype.map.call(
23           elms,(E)=>{r = new DOMObject(); r.elm=E;return r});
24     }
```

# 基本となるオブジェクト DOMObject–解説 (2)

- 17 行目から 19 行目で document.getElementById() に相当するこのオブジェクト用の関数を定義
- 20 行目から 24 行目で document.getElementsByTagName() の相当するこのオブジェクト用の関数を定義
  - 21 行目で指定された要素のリストを得ている。
  - 22 行目から 23 行目でそのリストを DOMObject のリストに変更
  - map は配列オブジェクト Array のメソッドで、各要素に対して引数で与えられた関数を実行し、その結果からなる配列を返す。
  - 21 行目で得たリストは配列ではないのでこのメソッドを使用不可
  - call は指定した関数が参照する this をその 1 番目の引数にする。
  - 23 行目の (E) 以降の書き方は新しい無名関数の記述方法。function(E){...}と書くのと同じ。

# 基本となるオブジェクト DOMObject—解説 (2 続き)

- 新しい記述一番の違いは関数内での this の取り扱い。
- class 内のコードは strict モードで実行
- function で定義された関数内では this は undefined
- 簡略化された記述では this の値がオブジェクト自身
- この違いが問題となる例はこのリストの 170 行目などにある。

# 基本となるオブジェクト DOMObject

```
25    constructor(elm, attribs, parentNode, events, nameSpace="HTML"){
26      if(elm){
27        this.elm = document.createElementNS(NS[nameSpace], elm);
28        this.setAttribs(attribs);//console.log(parentNode);
29        if(parentNode&& parentNode.elm) {
30          parentNode.elm.appendChild(this.elm);
31        }
32        for(let evt in events) {
33          this.elm.addEventListener(evt, events[evt], true);
34        }
35      }
36    }
37    setAttribs(Opt) {
38      let elm = this.elm;
39      for(let attrib in Opt) {
40        elm.setAttribute(attrib,Opt[attrib]);
41      }
42    }
43  };})();
```

# 基本となるオブジェクト DOMObject−解説

- 
- 25 行目から 43 行目はこのオブジェクトの constructor を定義している。
  - 引数は順に作成する要素名、その属性のリスト、親要素、イベント処理のリスト、名前空間 (デフォルトは HTML) である。
  - 27 行目で名前空間を指定して要素を作成している。
  - 28 行目は作成した要素に、与えられた属性を設定する関数を呼び出している。
  - 29 行目から 31 行目では親要素がある場合にはその子要素になるように指定している。
  - 32 行目から 34 行目ではイベント処理を設定している。
  - 37 行目から 42 行目では与えられたリストの属性を設定する関数を定義している。