ソフトウェア開発
第 13 回目授業

平野 照比古

2018/1/11

例の概要

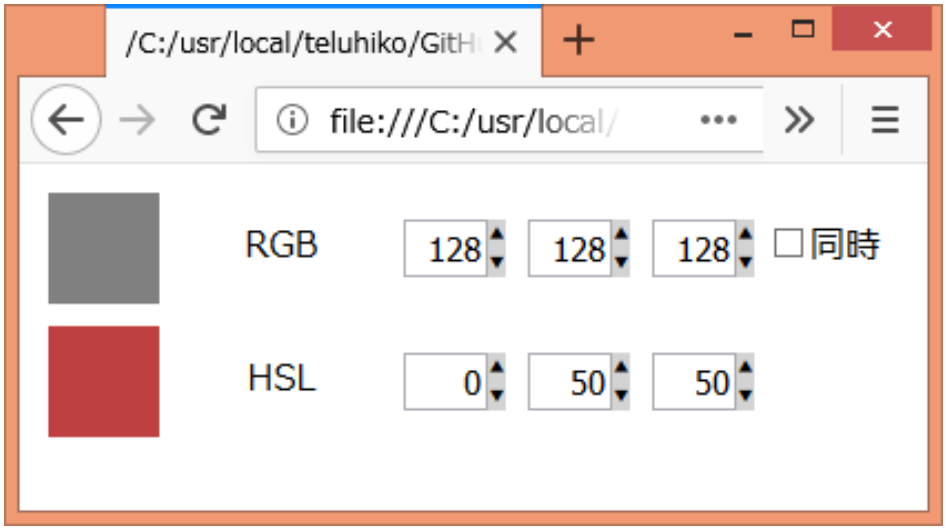

Figure: 色の指定を見る

- 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 {style: JSON.stringify(props).replace(/[{"]/g,"").replace(/[,}]/g,
9 }
```

- 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)

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

- 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 (35 行目の ())。
- 11 行目から 14 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。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      static getElmId(id) {
17        return {elm:document.getElementById(id)};
18      }
19      static getElmsTagName(elm) {
20        let elms = document.getElementsByTagName(elm);
21        return Array.prototype.map.call(
22            elms,(E)=>{r = new DOMObject(); r.elm=E;return r});
23      }
```

基本となるオブジェクト DOMObject–解説 (2)

- 16 行目から 18 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
- 19 行目から 23 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
  - 20 行目で指定された要素のリストを得ている。
  - 21 行目から 22 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。
  - 20 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。
  - 22 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 141 行目などにある。

基本となるオブジェクト DOMObject(3)

```
24      constructor(elm, attribs, parentNode, events, nameSpace="HTML"){
25        if(elm){
26          this.elm = document.createElementNS(NS[nameSpace], elm);
27          this.setAttribs(attribs);//console.log(parentNode);
28          if(parentNode&& parentNode.elm) parentNode.elm.appendChild(this.elm
29          for(let evt in events) this.elm.addEventListener(evt, events[evt],
30        }
31      }
```

基本となるオブジェクト DOMObject−解説 (3)

24 行目から 31 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)
- 26 行目で名前空間を指定して要素を作成
- 27 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。
- 28 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
- 29 行目ではイベント処理を設定
- 32 行目から 34 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

与えられた文字列を表示するためのクラス divWithText

```
36 class divWithText extends DOMObject{
37   constructor(text, parentNode, opt) {
38     super("div", opt, parentNode);
39     this.elm.innerText = text;
40   }
41 }
```

- このクラスは DOMObject を継承
- コンストラクタの引数は順に、表示するテキスト、親要素、属性の 3 つ
- 39 行目で 要素を作成し、その要素の innerText プロパティを設定することで表示を可能としている

SpinBox のクラス–constructor(1)

HTML にも spinbox があるが少し機能を変えている。
このクラスのコンストラクタの部分である。

```
42 class SpinBox {
43   constructor(Opt, P, Events, callback) {//console.log("called");
44     this.callback = callback;
45     this.values= {
46       max:Number.MAX_VALUE, min:-Number.MAX_VALUE,
47       skip: 1, bigSkip:5, value:0,
48       type:"limited"// "cyclic"
49     },
```

spinbox クラス–constructor 解説 (1)

- コンストラクタの引数は順に、オプション、親要素、イベント処理関数群、値が変化したときの処理関数
- 44 行目は値が変化したときに呼び出される関数をオブジェクトに登録
- 45 行目から 49 行目ではこのオブジェクトのデフォルトのパラメータを定義
  - max は設定できる値の最大値 (デフォルト値は JavaScript で扱うことができる最大値)
  - mix は設定できる値の最小値 (デフォルト値は JavaScript で扱うことができる最小値)
  - skip は値の変化量 (デフォルト値は 1)
  - bigSkip はシフトキーを押しながらクリックしたときの値の変化量 (デフォルト値は 5)
  - value は初期値 (デフォルト値は 0)
  - type は両端の値から外れたときの処理方法 (デフォルトは両端の値に固定–"limited"。値が循環的に変わる"cyclic"がある)

SpinBox のクラス–constructor(2)

```
50      this.boundary = {display:"block", margin: "0px", border: "0px", padding
51      let container = new DOMObject("div", setStyle(this.boundary, Opt), P);
52      setOptions(this.values, Opt);
53      let div0 = new DOMObject("div",
54        {style:"display:inline-block; vertical-align:middle;"},
55         container);
```

- 50 でオブジェクトの周りの余白の設定
- 51 行目で入れ物の 要素を作成
- 52 行目では 45 行目からのデフォルトのパラメータを与えられた値に変更
- 53 行目から 55 行目では 56 行目から 58 行目で定義されるテキストボックスを入れるための 要素を作成

SpinBox のクラス–constructor(2)

```
56      this.textBox = new DOMObject(
57        "input", {type:"text", size:"1em", value:this.values.value, readonly:
58             style:"font-size:15px; text-align:right; disable" }, div0);
59      let div = new DOMObject("div",
60            {style:"display:inline-block;vertical-align:middle"}, container);
61      let buttonStyle = {
62        style: "font-size:8px;margin:0px;cursor:default;"+
63               "background:lightgray;border-color:black;"
64      }
65      buttonStyle.type = "up";
66      this.buttonUP = new DOMObject(
67          "div", buttonStyle, div, {click:(E)=>{this.up = E.shiftKey;}});
68      this.buttonUP.elm.innerText ="▲";
69      buttonStyle.type = "down";
70      this.buttonDOWN = new DOMObject(
71          "div", buttonStyle, div, {click:(E)=>{this.down = E.shiftKey;}});
72      this.buttonDOWN.elm.innerText ="▼";
73    }
```

spinbox クラス–解説 (2)

- 56 行目から 58 行目で定義されるテキストボックスは値を直接変更できない設定をしている ("readonly"の指定)
- 59 行目から 60 行目ではテキストボックスの隣に置く上下のボタンを入れるための 要素を作成
- 61 行目から 64 行目では値の上下させるボタンの表示形式を定義
- 65 行目でさらにボタンのタイプ (値を増加) を設定し、66 行目から 67 行目でボタンとして機能するようなオブジェクトを追加。この上でクリックイベントが生じたときにイベント発生時ののシフトキーの状態を sipnbox の up プロパティの与えている (代入で行っているが、up は 76 行目から 85 行目で定義されているセッターである)。
- 69 行目から 72 行目は値を減少させるボタンを設定

セッターとゲッターを定義 (1)–値の増加の処理

```
74   get value()    {return this.values.value;}
75   set value(val) {this.textBox.elm.value = this.values.value = val;  }
76   set up(shift){
77       let skip = shift?this.values.bigSkip:this.values.skip;
78       this.value += skip;
79       if(this.values.type=="limited"){
80           this.value = Math.min(this.value, this.values.max);
81       } else {
82           if(this.value >=this.values.max) this.value -= this.values.max;
83       }
84       if(this.callback) this.callback();
85   }
```

セッターとゲッターを定義–解説

- 74 行目と 75 行目はそれぞれテキストボックスの値に関するゲッターとセッター
- 76 行目から 85 行目は設定値を増大させるメソッドであり、86 行目から 95 行目は設定値を減少させるメソッド
- 77 行目ではシフトキーが押されているかを判定して、変化量を決定
- 78 行目で仮の値を求め、それが上限値を超えている処理を 79 行目から 83 行目で行っている。
- 上限値で固定 (type が"limited") のときは上限値と仮の値の小さい方を設定 (80 行目)。
- 値が循環する (type が"cyclic") ときは上限値が超えた場合、上限値の値を引いている 821 行目)。
- 値の設定後の処理関数が定義されているときはその関数を呼び出す (84 行目)。

セッターとゲッターを定義–値の減少の処理

```
85    }
86    set down(shift){
87        let skip = shift?this.values.bigSkip:this.values.skip;
88        this.value -= skip;
89        if(this.values.type=="limited"){
90            this.value = Math.max(this.value, this.values.min);
91        } else {
92            if(this.value < this.values.min) this.value += this.values.max;
93        }
94      if(this.callback) this.callback()
95    }
96 }
```

値が減少する場合も同様

色の 3 成分を設定するクラス (1)

```
97 class SetColor {
98   constructor(title, type, P, callback, opt) {
99     //let that = this;
00     this.callback = callback;
01     if(title) {
02       new divWithText(title, P,
03         {style:"display:inline-block;width:100px;text-align:center"});
04     }
```

- コンストラクタに引数は順に、左端に示すテキスト、RGB 形式か HSL 形式のタイプ、親要素、値が変化したときの処理関数、オブジェクトのオプションである。
- 99 行目のコメント行は、メソッド内で呼び出される無名関数内でオブジェクト自身を示す this が参照できないときの対処法である。this を変数 (that) に格納し、それを参照する。
- 110 行目から 104 行目は左端にテキストが表示されるようにしている。

色の 3 成分を設定するクラス (2)

```
05      if(type.match(/^rgb$/i)) {
06        this.type = "rgb";
07        this.elm = new DOMObject("div",opt, P,
08            {click:(E)=>{
09            if(this.checkBox.elm.checked){
10              this.RGB.forEach((C)=>
11                {C[E.target.getAttribute("type")] = E.shiftKey;});
12              E.stopPropagation();
13              if(this.callback) this.callback(E);
14            }
15          }
16        }).elm;
```

色の 3 成分を設定するクラス–タイプ"rgb"解説 (1)

- 106 行目でタイプを"rgb"に設定
- 107 行目から 116 行目でオブジェクト全体を囲む要素を作成して、そこに click イベントの処理関数を定義
- 右端のチェックボックスにチェックがある場合は RGB の値を一斉に増減。そのときは (109 行目) この関数で処理 (110 行目から 112 行目)。
- 110 行目から 111 行目で RGB の値を変化 (this.RGB.forEach)。
- それぞれの sipnbox にもイベントの処理関数がついているので、2 回処理させないために、112 行目でイベントの伝搬を止める (E.stopPropagation())。
- その後、登録された処理関数を呼び出している。

色の 3 成分を設定するクラス–タイプ"rgb"(2)

```
17      this.RGB = Array(3).fill(0).map(()=>{
18        return new SpinBox({
19          type:"limited",
20          display:"inline-block", margin:"5px",
21          value:128, max:255, min:0
22          },
23          this, {}, this.callback);
24      });
25      this.checkBox = new DOMObject("input",
26          {type:"checkbox", style:"display:inline-block"},this);
27      new divWithText("同時", this, {style:"display:inline-block;"});
28    } else {
```

色の 3 成分を設定するクラス–タイプ"rgb"解説 (2)

- 126 行目から 133 行目では spinbox3 つ作成のために、大きさ 3 の配列を作成し (new Array(3))、それぞれに 0 を設定 (fill(0)) する。この配列に forEach を用いて spinbox を作成
  Array のメソッド forEach は undefined の要素に対して実行されないのでこのような処理が必要となる。
- これらの spinbox はタイプが"limited"、最小値が 0、最大値が 255、初期値を 128 である。
- 134 行目から 136 行目では RGB 値を同時に変化させるかどうかのチェックボックスを作成

色の 3 成分を設定するクラス–タイプ"rgb"(2)

```
29        this.type = "hsl";
30        this.elm = new DOMObject("div",opt, P).elm;
31        this.HSL = [[360,0],[100,50],[100,50]].map((V)=>{
32          return new SpinBox({
33            type:"limited",
34            display:"inline-block", margin:"5px",
35            value:V[1], max:V[0], min:0
36            },
37            this, {}, this.callback);
38        });
39        this.HSL[0].values.type="cyclic";
```

色の 3 成分を設定するクラス–タイプ"hsl"

- 129 行目でタイプを"hsl"に設定している。
- 130 行目ではオブジェクト全体を囲む要素を定義している。
- 131 行目から 1138 目では 3 つの spinbox を定義している。
- 色相 (H) と残りの 2 つでは値の範囲と取り扱いが異なるので、範囲と初期値を与える配列を利用して 3 つの spinbox を作成している。
- 139 行目でははじめの spinbox のタイプを"cyclic"に修正している。

色の 3 成分を設定するクラス (3)

```
40      }
41    }
42    get rgb()    {return this.RGB.map((V)=>{return V.value});}
43    set rgb(vals){vals.forEach((V, i)=>{this.RGB[i].value = V;});}
44    get hsl()    {return this.HSL.map((V)=>{return V.value});}
45    set hsl(vals){vals.forEach((V, i)=>{this.HSL[i].value = V;});}
46    toString(){
47      return (this.type == "rgb") ?
48        `rgb(${this.rgb.join(",")})`:
49        `hsl(${this.hsl[0]},${this.hsl[1]}%,${this.hsl[2]}%)`;
50    }
51  }
```

色の 3 成分を設定するクラス (3)–解説

- 142 行目では rgb の値をコピーして配列で返すゲッターを、143 行目では rgb の値のセッターをを定義している。
- 144 行目では hsl の値をコピーして配列で返すゲッターを、145 行目では hsl の値のセッターをを定義している。
- 146 行目から 150 行目では CSS3 で定義されている RGB や HSL 形式に変換するメソッドを定義している。

色の変化を見えるようにする

```
1 window.onload = function(){
2   let areaColor = ["RGB", "HSL"].map((type)=>{
3     let area = DOMObject.getElmId(type);
4       return [area, new SetColor(type, type, area, function(){setColors();}
5                   {style:"display:inline-block;vertical-align:middle;"})];
6     });
7   setColors();
8   function setColors(){
9     areaColor.forEach((C)=>
10      {C[0].elm.children[0].style.background = `${C[1]}`;});
11  }
12 }
```

色の変化を見えるようにする–(解説)

- 2 行目から 6 行目までで 2 つの色が設定できるオブジェクトを作成している。一つ目は RGB 形式で 2 つ目が HSL 形式となる。区別するパラメータは配列で与えている。
- 7 行目ではその値を左側の背景色に反映するための関数を呼び出している。
- 8 行目から 11 行目では spinbox で設定された色を背景色に設定する関数である。
- 10 行目で spinbox の値を文字列に変換することで行っている。