

Web ページ上のカラーユニバーサルデザインに関する実験

1250373 溝口 洋熙 *

Group 10

June 7th, 2023

概要

この実験では、Web ページ上のコンテンツを、カラーユニバーサルデザインに則したもので作成する。まず初めに、必要な HTML, CSS, JavaScript の基本的な扱いを簡易的な実験で示す。それらの技術を応用して、表、円グラフ、折れ線グラフを、カラーユニバーサルデザインに則して作成する。また、一般色覚者と色覚障害者の見え方を示し、カラーユニバーサルデザインの適用方法と、色覚障害者の見え方を色覚特性とともに考察する。この実験を通して、色覚特性の違いや、色覚特性を根拠とした見え方の違いが明らかになった。色覚障害者でも一般色覚者と同等のサービスを享受できる設計について明らかになった。

1 実験の目的

本実験では、Web ページ上のコンテンツを、一般色覚者と色覚障害者が相違なく利用できるように、カラーユニバーサルデザインに則して設計する。Web ページの設計にあたって、CSS, DOM, JavaScript の技術を確認するために、いくつかのコンテンツを作成する。その技術を用いて、カラーユニバーサルデザインに則した Web ページを作成する。今回作成するコンテンツは、表、折れ線グラフ、円グラフである。それぞれのコンテンツに対して、一般色覚者、色覚障害者の見え方を結果として示し、カラーユニバーサルデザインで必要な要素について考察する。

2 技術要素と実験の方法

2.1 装置と用語解説

この実験には MathWorks® 社の MATLAB® を用いて、表 1 の環境下で実験する。

表 1: 実験環境

実験機	MacBook Air 2022 (Apple 社) 型番: MLY13J/A
プロセッサ	Apple Silicon M2 8 コア CPU, 8 コア GPU
メモリ	8GB
ブラウザ	Safari バージョン 16.5
ImageJ	1.53k Java 13.0.6 (64-bit)

■HTML Hyper Text Markup Language の略で、Web ページを作成するためのマークアップ言語。Web ページ上のテキストデータに「タグ」を与えて、文字の大きさ、色やフォントを変更する。

■CSS Cascading Style Sheets の略で、Web ページのスタイルを指定するための言語である。CSS の適用により、ホームページの文字や背景などがタグを利用して統一される。

■JavaScript ブラウザ上で動作するアプリケーションを記述するための言語である。現在の Web アプリケーション上でデファクトスタンダード言語である [1, p.68]。

2.2 JavaScript の初步

• A.indexOf("Strings")

JavaScript には、A.indexOf("Strings") で、Strings が A に含まれていることを確認できる。含まれていない場合は、-1 を戻り値とし、含まれている場合はその箇所を自然数で返す。

• User Agent (UA)

User Agent (UA) とは、利用者のブラウザと OS を指す。この実験で利用する UA は、macOS の Safari である。Safari には UA を変更する機能がある。Chrome, Firefox, MicrosoftEdge, Safari 16.4 は、Safari の UA 切り替え機能を用いて実験する。

► ブラウザ判定

Navigator object オブジェクトの userAgent プロパティを用いて、利用ブラウザを判定する。A に Strings が含まれていない場合、戻り値は -1 である。この実験では、ブラウザが Firefox である場合は this browser is Firefox

*高知工科大学 情報学群 情報セキュリティシステム研究室

、そうでない場合は this browser is not Firefox と表示する Web ページを作成する。

► ブラウザによる CSS の切り替え

ブラウザ判定を用いて、ブラウザによって CSS を変更する JavaScript を記述する。CSS を設定する link タグに id を指定し、document.getElementById 関数で link タグをインスタンス化する。ブラウザ判定結果により、このインスタンスを利用して CSS を変更する。この実験では、CSS で背景色のみを変更する。UA と指定背景色は表 2 に示す。

表 2: UA と指定背景色

UA	背景色	CSS 名
Firefox	■ orange	firefox.css
Chrome	■ blue	chrome.css
Other	■ gray	default.css

ここで、MicrosoftEdge は、UA に"Chrome"文字列と"Edge"文字列を持つ。正確に Chrome を判別するには、条件式に、「"Chrome"を含み"Edge"を除く」処理を記述する必要がある。

► 時刻による CSS の切り替え

JavaScript で現在時刻を取得するには、Date をインスタンス化する必要がある。Date 内の getSeconds 関数を呼び出すことで、現在時刻の「秒」を得られる。この実験では、現在時刻*の秒数を n とすると、以下の条件で CSS を変更する。

$$\text{適用する CSS} = \begin{cases} \text{firefox.css} & (0 \leq n < 20) \\ \text{chrome.css} & (20 \leq n < 40) \\ \text{default.css} & (40 \leq n < 60) \end{cases}$$

► マウスイベントの取得

この実験では、「現在時刻の取得」と書かれた文字上をクリックすると、クリックした時刻をリストとして後ろに追加するプログラムを記述する。文字列「現在時刻の取得」をひとつのオブジェクトとして定義するため、`` タグを用いる。また、リストアイテムとして表示するために、現在時刻を`` タグ内の要素`` タグへ格納する。

- 「現在時刻の取得」文字列を`<p></p>` タグの属性 `onmousedown` で、現在時刻をリストアイテムとして追加する関数 `input_item()` を呼び出す。この関数を呼び出すと、変数 `time` に、時刻 `Hour:Minutes:Second` が格納される。

*正確には、HTML を読み込んだ時刻。

- ``に対して、`document.getElementById` 関数を用いてインスタンス名 `list` でインスタンス化する。
- `document.createElement("li")` 関数を用いて、`` タグを生成し、変数 `listItem` に格納する。
- `listItem` の要素を `listItem.textContent` で指定し、`time` を代入する。
- `list.appendChild(listItem)` で `listitem` を子要素として組み込む。

2.3 色覚

色覚は、光の刺激により色を見分ける感覚である [2]。すべての色は、光の三原色と呼ばれる「赤」「緑」「青」から構成される。色を感じる視細胞も、赤緑青それぞれに敏感な細胞が 3 種類存在する。色覚異常はこの 3 種類の視細胞のうちどれか足りなかったり、十分に機能しないために起こる [3]。各色覚の見え方を表 4 に示す。

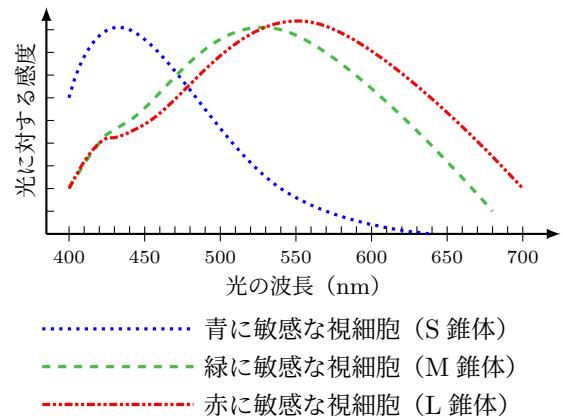


図 1: 各視細胞の光に対する感度（概略）[4]

表 3: 色覚と状態

色覚	状態
C 型色覚	3 錐体が正常。一般色覚。
P 型色覚	L 錐体に異常がある。赤色と灰色の識別が困難。
D 型色覚	M 錐体に異常がある。緑色と灰色の識別が困難。
T 型色覚	S 錐体に異常がある。青色と黄色が混同。

表 4: 各色覚の見え方 [5]

色覚	色の見え方
C 型色覚	
P 型色覚	
D 型色覚	
T 型色覚	

表 4 より、4種の色覚に対応する色を決定する。この実験では、講義一覧、折れ線グラフ、円グラフを4つの色覚に対応できるように設計する。各コンテンツに対して、各色覚での見え方を結果として出力する。出力には、ImageJアプリケーションの VischeckJ1 プラグインを利用する。VischeckJ1 は、イメージに対して、P型、D型、T型色覚での見え方を表示するプラグインである。

▶ 講義一覧

HTML の `table` タグを用いて、講義一覧を作成する。各講義の「専門基礎科目」、「専門発展科目」、「専門領域科目」の背景色と、履修登録必須科目の文字色を、CSS で設定する。デフォルトのセル背景配色を表 5 に示す。また、履修登録必須科目は、`<p1></p1>` タグ内の `color` 属性で指定し、色を `#ff0000` とする。

表 5: デフォルト配色

分類	セル背景色	class 名
専門基礎科目	■	kiso
専門発展科目	■	hattent
専門領域科目	■	ryouiki

表 6: カラーユニバーサルデザイン配色

分類	セル背景色	class 名
専門基礎科目	■	kiso
専門発展科目	■	hattent
専門領域科目	■	ryouiki

■ `#b4ebfa` は、■ `#ffd1d1`、■ `#ffff99` は、RGB 三原色を多く含んでいる。この3色の組み合わせは、3原色の中で1つの認識が困難でも、ほかの2色で識別が可能である。これらは白に近い淡い色であり、文字色を `black` にすることで、色覚によらず文字をはっきりと認識できる。履修登録必須科目は、色に依存しないように文字色を変えず、文字を太くし、下線を引くことで表現した (src.1)。

src. 1: 太字と下線を設定する CSS

```
p1 {
  font-weight: bold;
  text-decoration: underline;
}
```

▶ 折れ線グラフ

JavaScript で記述された折れ線グラフを、カラーユニバーサルデザインに従って変更する。以下の点を変更する。

- ・背景の灰色グラデーションを削除する。
- ・線の太さを太くする。
- ・マークの形状を商品別に変更する。
- ・線の配色を変更する。
 - ■ `#0080ff`, ■ `#bf00bf` は色覚によらず色が見える。
 - ■ `#ff0000` は P型、D型色覚には異なった色に見えるが、ほかの2つと重複しない。
 - 今回は、線自体が値を示すので、淡い色ではなく、濃い色を採用した。

マーク形状、マークと線の太さ、背景色と線色の変更は src.2 で行う。

src. 2: 折れ線グラフの変更

```
var parms = {
  // 略
  // 背景色の変更
  backgroundColor: "#ffffff",
  gbackgroundColor: "#ffffff",
  lineWidth: 3, // ラインの太さ
  dotRadius: 4, // マークの太さ
  // マークの形状
  dotType: ["diamond", "square", "disc"],
  // 略
}
// 略
var colors = ["0,128,255", "255,0,0", "191,191,191"]; // 線色の設定
```

▶ 円グラフ

- ・円グラフの配色を変更する。以下の配色を、いずれの色覚でも似た色どうしが隣り合わないように配置する。
 - ■ `#ff0000`
 - ■ `#4169ff`
 - ■ `#ffff00`
 - ■ `#00ffff`

円グラフの配色を変更は src.3 で行う。

src. 3: 円グラフの変更

```
var colors = ["255,0,0", "65,105,255", "255,255,0", "0,255,255"]; // 円色の変更
```

3 実験の結果

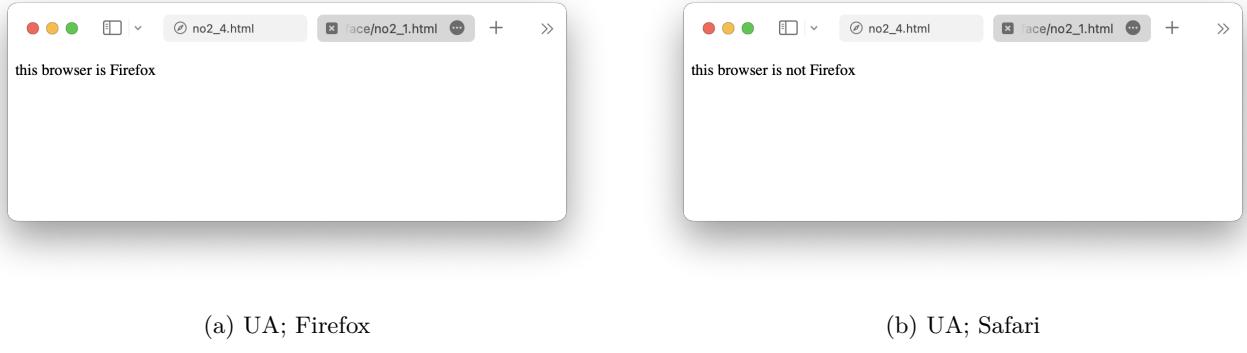


図 2: ブラウザ判定



図 3: ブラウザ判定による CSS の変更

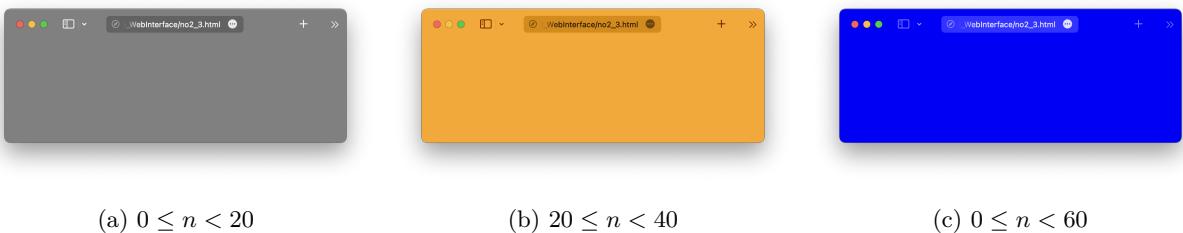


図 4: 現在時刻秒数 n に対する CSS の切り替え

「現在時刻」文字列をクリックすると、現在時刻のリストアイテムが下部に追加された。

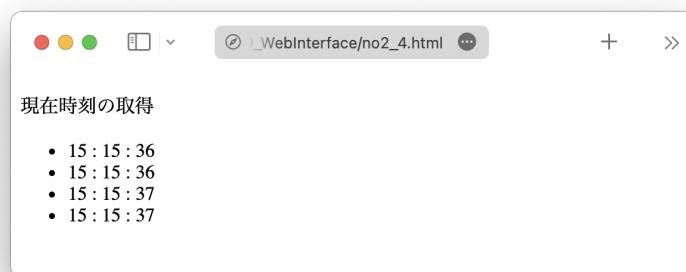


図 5: マウスイベントの取得

1年次	2年次	3年次	4年次	1年次	2年次	3年次	4年次
情報学群ガイダンス	計算機言語	ソフトウェア工学	情報学群特別講義	情報学群ガイダンス	計算機言語	ソフトウェア工学	情報学群特別講義
応用CL	情報理論基礎	ソフトウェア工学演習	プロ研	応用CL	情報理論基礎	ソフトウェア工学演習	プロ研
数学1	情報学群実験第1	キャリアプラン1	卒業研究	数学1	情報学群実験第1	キャリアプラン1	卒業研究
物理学概論	キャリアプラン基礎	キャリアプラン2		物理学概論	キャリアプラン基礎	キャリアプラン2	

(a) 改良前 (C型)

1年次	2年次	3年次	4年次
情報学群ガイダンス	計算機言語	ソフトウェア工学	情報学群特別講義
応用CL	情報理論基礎	ソフトウェア工学演習	プロ研
数学1	情報学群実験第1	キャリアプラン1	卒業研究
物理学概論	キャリアプラン基礎	キャリアプラン2	

(b) 改良後 (C型)

1年次	2年次	3年次	4年次
情報学群ガイダンス	計算機言語	ソフトウェア工学	情報学群特別講義
応用CL	情報理論基礎	ソフトウェア工学演習	プロ研
数学1	情報学群実験第1	キャリアプラン1	卒業研究
物理学概論	キャリアプラン基礎	キャリアプラン2	

(c) 改良前 (P型)

1年次	2年次	3年次	4年次
情報学群ガイダンス	計算機言語	ソフトウェア工学	情報学群特別講義
応用CL	情報理論基礎	ソフトウェア工学演習	プロ研
数学1	情報学群実験第1	キャリアプラン1	卒業研究
物理学概論	キャリアプラン基礎	キャリアプラン2	

(d) 改良後 (P型)

1年次	2年次	3年次	4年次
情報学群ガイダンス	計算機言語	ソフトウェア工学	情報学群特別講義
応用CL	情報理論基礎	ソフトウェア工学演習	プロ研
数学1	情報学群実験第1	キャリアプラン1	卒業研究
物理学概論	キャリアプラン基礎	キャリアプラン2	

(e) 改良前 (D型)

1年次	2年次	3年次	4年次
情報学群ガイダンス	計算機言語	ソフトウェア工学	情報学群特別講義
応用CL	情報理論基礎	ソフトウェア工学演習	プロ研
数学1	情報学群実験第1	キャリアプラン1	卒業研究
物理学概論	キャリアプラン基礎	キャリアプラン2	

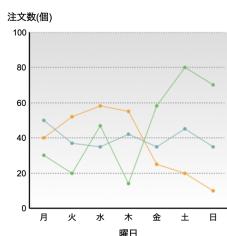
(f) 改良後 (D型)

1年次	2年次	3年次	4年次
情報学群ガイダンス	計算機言語	ソフトウェア工学	情報学群特別講義
応用CL	情報理論基礎	ソフトウェア工学演習	プロ研
数学1	情報学群実験第1	キャリアプラン1	卒業研究
物理学概論	キャリアプラン基礎	キャリアプラン2	

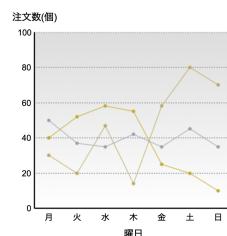
(g) 改良前 (T型)

(h) 改良後 (T型)

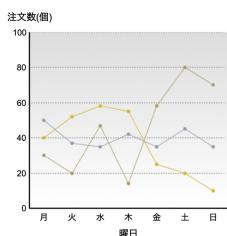
図 6: 講義一覧



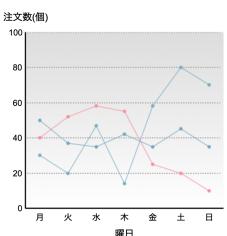
(a) 改良前 (C型)



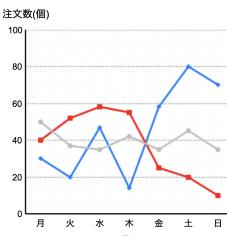
(b) 改良前 (P型)



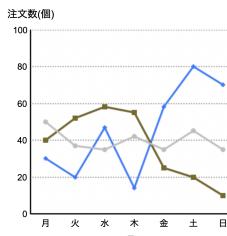
(c) 改良前 (D型)



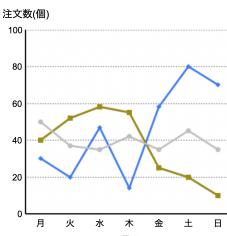
(d) 改良前 (T型)



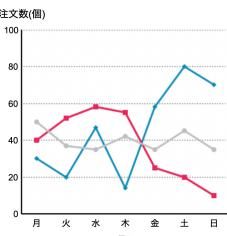
(e) 改良後 (C型)



(f) 改良後 (P型)

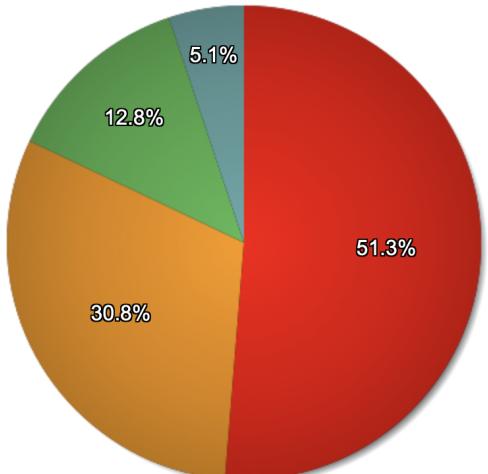


(g) 改良後 (D型)

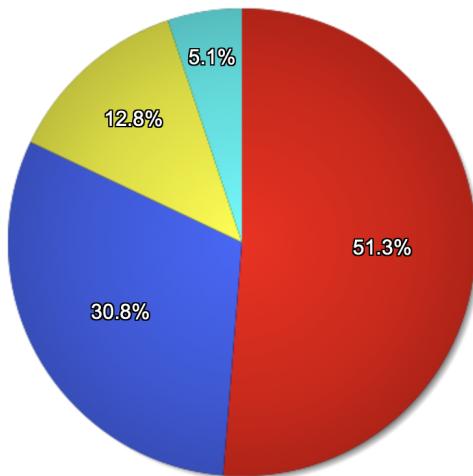


(h) 改良後 (T型)

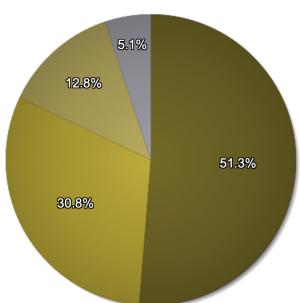
図 7: 折れ線グラフ



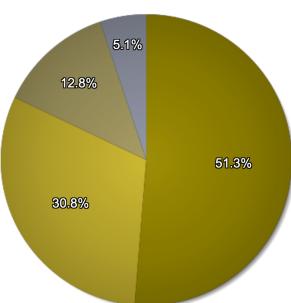
(a) 改良前 (C型)



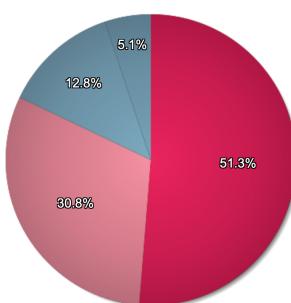
(b) 改良後 (C型)



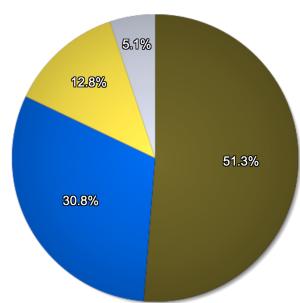
(c) 改良前 (P型)



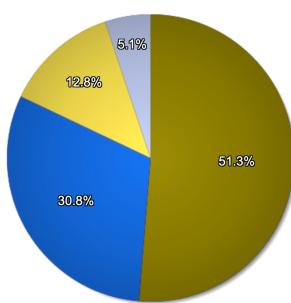
(d) 改良前 (D型)



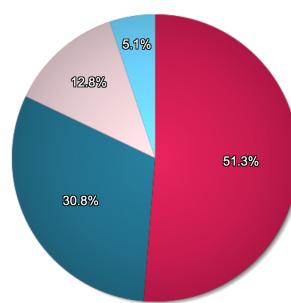
(e) 改良前 (T型)



(f) 改良後 (P型)



(g) 改良後 (D型)



(h) 改良後 (T型)

図 8: 円グラフ

4 考察

講義一覧、折れ線グラフ、円グラフのコンテンツを、すべての色覚で内容を判別できるように配色した。折れ線グラフについては、マークの形状を商品別に変更することで、商品を区別した。マークは線と同化し、一部見にくいで、線の形状やマークの色を変更することで、さらに見やすいデザインになる。また、円グラフは、色の変更だけでなく、項目間の感覚を開けることで、色の境界に対して見やすさを保証できる。

■色覚 一般色覚(C型色覚)の人々は、L(赤)錐体、M(緑)錐体、S(青)錐体の3種類を持っており、すべて正常に働いている。色弱者(色の配慮が不十分な社会における弱者)は、いずれかの錐体がない、またはいずれかの錐体が不十分な働きである。たとえば、M錐体がない、または不十分な働きであるD型色覚では、図9のように2つの周波数領域で混同が生じる。

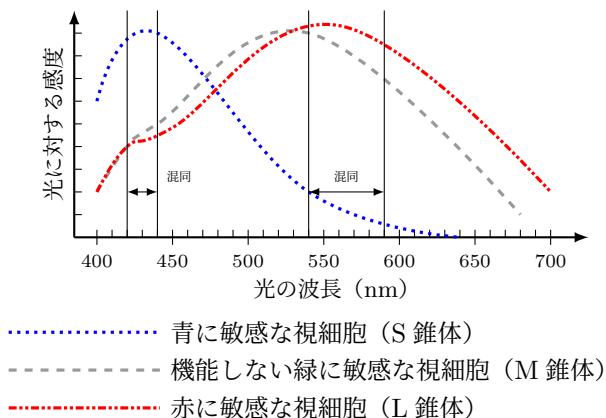


図9: D型色覚の周波数に対する感度(概略) [6]

C型色覚の場合は、L錐体、M錐体、S錐体それぞれからの入力に対して比を取り、色を弁別している。それに対して、D型色覚の場合は、L錐体とS錐体の入力の比で色を弁別しているため、L錐体とS錐体の比が同じ箇所は、色の区別がつかないと考えられる。この混同部分を避けるために、L錐体とS錐体の感度比が異なる部分を採用する必要があるだろう。

[5]

5 結論

この実験を通して、色覚特性を根拠とした、色覚障害者の見え方の違いについて明らかになった。私自身は一般色覚を持っており、たとえば私は「赤色」を■redと認識できる。しかし■redを誰しも認識できる前提で、重要なことがら(避難ルートや警報)を伝える看板や掲示板を設計すると、色弱者は文字や意味を読み取れずに命を落としかねない。

い。カラーユニバーサルデザインは、色弱者は一般色覚を持つ人と平等なサービスを享受するために必要なデザインである。

このレポートでは、カラーユニバーサルデザインに配慮した設計となっている。色を示すときに用いたカラー・ボックス(■orange)や、線グラフの形状がその例である。

6 関連語句

6.1 DOM

DOMとは、Document Object Modelの略称で、プログラムから構造化された文書を扱うためのモデルである。DOMは、オブジェクトを操作するAPIを提供している。JavaScriptなどのいろいろなプログラミング言語から利用できる。DOMは、データを木構造で扱う。この構造の構成要素をノード(Node)という。構造化された文書の例としてXMLをあげる。例として、以下のようなXML文書に対して、DOM treeで表現する。

src. 4: XML文書

```
<?xml version="1.0" encoding="utf-8"?>
<a version="2.0">
  <b>
    <c>
      <d>KUT</d>
      <f>kut.com</f>
    </c>
  </b>
</a>
```

```
| - Document
  `-- Element: a version="2.0"
    `-- Element: b
      `-- Element: c
        | - Element: d
        |   '-- Text: KUT
        `-- Element: f
          '-- Text: kut.com
```

図10: DOM tree

Documentは文書全体を表すノード、Elementは要素を表すノード、Textはテキストを表すノードである。ここで、Textも1つのノードとなることに注意したい。

[7]

6.2 WebGL

OpenGL(Open Graphics Library)とは、3次元グラフィックスライブラリである。これを用いることで、高品質な仮想空間を表現できる。構成要素として、3次元立体

や 2 次元画像が挙げられる [8]. WebGL とは、ブラウザ上で 3 次元グラフィックを実現する技術である。HTML5 の Canvas 要素に対して、ブラウザで JavaScript などを使用して OpenGL で描画する。WebGL は、ブラウザだけで、コンテンツの 3 次元描画が可能になり、アプリケーションのインストールやバージョンアップは必要なく、OpenGL を用いるため、Web 上でネイティブアプリケーションと同等のパフォーマンスを再現できる。

[9]

6.3 three.js

three.js は、JavaScript ベースの WebGL エンジンである。グラフィックスを駆使したアプリケーションをブラウザから直接実行できる。three.js ライブライリは、ブラウザ上で 3D 描画するための機能と API を提供している。WebGL と three.js は、API かライブラリであるかの違いである。WebGL はブラウザ上 3D グラフィックスをレンダリングするための JavaScript API であるのに対し、three.js は、WebGL 上で 3D グラフィックスを作成するためのライブラリである。three.js は、物理演算や光源の位置により物体の光り方を変更できたりなど、数学やプログラミングの知識がなくとも簡単に 3D グラフィックスを作成できる。

[10]

参考文献

- [1] 三上浩司, 渡辺大地. CG とゲームの技術（メディア学大系）. メディア学大系. コロナ社, 2016.
- [2] 新明解国語辞典. 三省堂, 2012.
- [3] 株式会社三和化学研究所. 色覚の異常. https://www.skk-net.com/health/me/c01_13.html, Confirmation date: May 28th, 2023.
- [4] 藤枝市. 視覚情報のためのカラーユニバーサルデザインガイドライン. <https://www.city.fujieda.shizuoka.jp/material/files/group/109/Coloruniversaldesignguideline.pdf>, Confirmation date: May 29th, 2023.
- [5] 秀潤社. 色覚が変化すると、どのように色が見えるのか？～赤緑色盲の人にはどのように色が見えるのか～. <https://www.nig.ac.jp/color/barrierfree/barrierfree2-2.html>, Confirmation date: May 28th, 2023.
- [6] 秀潤社. 色覚が変化すると、どのように色が見えるのか？～ヒトの色覚と先天色盲について～. <https://www.nig.ac.jp/color/barrierfree/barrierfree2-1.html>, Confirmation date: May 28th, 2023.
- [7] 東京電機大学. XML 文書と DOM. <https://www.mlab.im.dendai.ac.jp/~yamada/web/xml/dom/>, Confirmation date: May 29th, 2023.
- [8] 吉本 富士市教授, 呉 海元助教授. Open GL 入門. <http://web.wakayama-u.ac.jp/~wuhy/GSS/01.html>, Confirmation date: May 29th, 2023.
- [9] みずほリサーチ&テクノロジーズ. WebGL を用いた三次元表示 web アプリケーション開発. <https://www.mizuho-rt.co.jp/solution/research/telecom/system/webgl/index.html>, Confirmation date: May 29th, 2023.
- [10] moz://a. three.js. https://developer.mozilla.org/en-US/docs/Glossary/Three_js, Confirmation date: May 29th, 2023.

付録

A WEB インターフェイス (May 22th, 2023)

src. A-1: 講義一覧の作成

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <meta charset="utf-8">
6   <title>Title</title>
7   <link rel="stylesheet" href="no1.css">
8 </head>
9
10 <body>
11   <h1>講義一覧表</h1>
12   <h3>1250373 溝口 洋熙</h3>
13   <h3>May 22th, 2023</h3>
14   <table border="1">
15     <tr>
16       <th>1年次</th>
17       <th>2年次</th>
18       <th>3年次</th>
19       <th>4年次</th>
20     </tr>
21     <tr>
22       <td class="kiso">
23         <p1>情報学群ガイダンス</p1>
24       </td>
25       <td class="kiso">
26         <p1>計算機言語</p1>
27       </td>
28       <td class="hatten">
29         <p1>ソフトウェア工学</p1>
30       </td>
31       <td class="ryouiki">情報学群特別講義</td>
32     </tr>
33     <tr>
34       <td class="kiso">
35         <p1>応用CL</p1>
36       </td>
37       <td class="kiso">
38         <p1>情報理論基礎</p1>
39       </td>
40       <td class="hatten">
41         <p1>ソフトウェア工学演習</p1>
42       </td>
43       <td class="ryouiki">プロ研</td>
44     </tr>
45     <tr>
46       <td>数学1</td>
47       <td class="ryouiki">
48         <p1>情報学群実験第1</p1>
49       </td>
50       <td>キャリアプラン1</td>
```

```

51      <td class="ryouiki">卒業研究</td>
52  </tr>
53  <tr>
54      <td>物理学概論</td>
55      <td>キャリアプラン基礎</td>
56      <td>キャリアプラン2</td>
57      <td></td>
58  </tr>
59  </table>
60 </body>
61
62 </html>

```

src. A-2: 講義一覧の作成 CSS

```

1  table {
2      text-align: center;
3  }
4
5 .kiso {
6     background-color: #55bb55;
7 }
8
9 .hattan {
10    background-color: #ff9900;
11 }
12
13 .ryouiki {
14     background-color: #66aaaa;
15 }
16
17 p1 {
18     color: #ff0000;
19 }

```

src. A-3: ブラウザ判定

```

1  <!DOCTYPE html>
2  <html>
3
4  <head>
5
6  </head>
7  <script type="text/javascript">
8      if (navigator.userAgent.indexOf("Firefox") != -1) {
9          document.write("this browser is Firefox"); document.close();
10     } else {
11         document.write("this browser is not Firefox"); document.close();
12     } </script>
13
14 <body>
15
16 </body>
17
18 </html>

```

src. A-4: ブラウザによるスタイルシートの切り替え

```
1 <!DOCTYPE html>
2 <html>
3 <link rel="stylesheet" href="no1.css" id="CSS">
4
5 <head>
6
7 </head>
8 <script type="text/javascript">
9     var elem = document.getElementById("CSS");
10    var userAgent = navigator.userAgent;
11    if (userAgent.indexOf("Firefox") != -1) {
12        elem.href = "no2_2_firefox.css";
13    } else if (userAgent.indexOf("Chrome") != -1 && userAgent.indexOf("Edg")) {
14        elem.href = "no2_2_chrome.css";
15    } else {
16        elem.href = "no2_2_default.css";
17    }
18 </script>
19
20 <body>
21
22 </body>
23
24 </html>
```

src. A-5: 時刻によるスタイルシートの切り替え

```
1 <!DOCTYPE html>
2 <html>
3 <link rel="stylesheet" href="no1.css" id="CSS">
4
5 <head>
6
7 </head>
8 <script type="text/javascript">
9     const agent = window.navigator.userAgent.toLowerCase();
10    var elem = document.getElementById("CSS");
11    var date = new Date();
12    var sec = date.getSeconds();
13    var userAgent = navigator.userAgent;
14    if (0 <= sec && sec < 20) {
15        elem.href = "no2_2_firefox.css";
16    }
17    else if (20 <= sec && sec < 40) {
18        elem.href = "no2_2_chrome.css";
19    } else if (40 <= sec && sec <= 59) {
20        elem.href = "no2_2_default.css";
21    }
22 </script>
23
24 <body>
25
26 </body>
27
```

28 | </html>

src. A-6: マウスイベントの取得

```
1 <!DOCTYPE html>
2 <html>
3 <link rel="stylesheet" href="no1.css" id="CSS">
4
5 <head>
6
7 </head>
8 <script type="text/javascript">
9     function input_item() {
10         var date = new Date();
11         var sec = date.getSeconds();
12         var min = date.getMinutes();
13         var hour = date.getHours();
14         var st = " : ";
15         var time = String(hour) + st + String(min) + st + String(sec)
16
17         var list = document.getElementById("imz");
18         var listItem = document.createElement("li");
19         listItem.textContent = time;
20         list.appendChild(listItem);
21     }
22 </script>
23
24 <body>
25     <span>
26         <p onmousedown="input_item()">現在時刻の取得</p>
27     </span>
28     <ul id="imz">
29         </ul>
30 </body>
31
32 </html>
```

src. A-7: default.css

```
1 body {
2     background-color: gray;
3 }
```

src. A-8: chrome.css

```
1 body {
2     background-color: blue;
3 }
```

src. A-9: firefox.css

```
1 body {
2     background-color: orange;
3 }
```

src. A-10: マウスイベントの取得

```
1 <!DOCTYPE html>
2 <html>
3 <link rel="stylesheet" href="no1.css" id="CSS">
4
5 <head>
6
7 </head>
8 <script type="text/javascript">
9     function input_item() {
10         var date = new Date();
11         var sec = date.getSeconds();
12         var min = date.getMinutes();
13         var hour = date.getHours();
14         var st = " : ";
15         var time = String(hour) + st + String(min) + st + String(sec)
16
17         var list = document.getElementById("imz");
18         var listItem = document.createElement("li");
19         listItem.textContent = time;
20         list.appendChild(listItem);
21     }
22 </script>
23
24 <body>
25     <span>
26         <p onmousedown="input_item()">現在時刻の取得</p>
27     </span>
28     <ul id="imz">
29     </ul>
30 </body>
31
32 </html>
```

B カラーユニバーサルデザイン (May 25th, 2023)

src.A-1 に src.B-1 を適用する。

src. B-1: 講義一覧 CSS 改良版

```
1 table {
2     text-align: center;
3 }
4
5 .kiso {
6     background-color: #B4EBFA;
7 }
8
9 .hattan {
10    background-color: #FFD1D1;
11 }
12
13 .ryouiki {
14    background-color: #FFFF99;
15 }
16
17 p1 {
```

```
18     font-weight: bold;
19     text-decoration: underline;
20 }
```

src. B-2: 折れ線グラフ HTML 改良版

```
1 <!DOCTYPE html>
2 <html lang="ja">
3
4 <head>
5   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
6   <script type="text/javascript" src="./no2_line.js"></script>
7   <script type="text/javascript">
8     window.onload = function () {
9       var lg = new html5jp.graph.line("graph01");
10      if (!lg) {
11        return;
12      }
13      var items = [[["商品A", 30, 20, 47, 14, 58, 80, 70], ["商品B", 40, 52, 58, 55, 25, 20,
14        10], ["商品C", 50, 37, 35, 42, 35, 45, 35]]];
15      var params = {
16        x: ["曜日", "月", "火", "水", "木", "金", "土", "日"],
17        y: ["注文数(個)", 0, 20, 40, 60, 80, 100],
18        yMax: 100,
19        yMin: 0,
20        dLabel: false,
21      };
22      lg.draw(items, params);
23    };
24  </script>
25 </head>
26
27 <body>
28
29   <div><canvas width="600" height="450" id="graph01"></canvas></div>
30
31 </body>
32
33 </html>
```

src. B-3: 折れ線グラフ JavaScript 改良版

```
1 // Copyright 2007-2009 futomi  http://www.html5.jp/
2 //
3 // Licensed under the Apache License, Version 2.0 (the "License");
4 // you may not use this file except in compliance with the License.
5 // You may obtain a copy of the License at
6 //
7 //   http://www.apache.org/licenses/LICENSE-2.0
8 //
9 // Unless required by applicable law or agreed to in writing, software
10 // distributed under the License is distributed on an "AS IS" BASIS,
11 // WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 // See the License for the specific language governing permissions and
13 // limitations under the License.
```

```

14 // 
15 // line.js v1.0.2
16 
17 if (typeof html5jp == 'undefined') {
18     html5jp = new Object();
19 }
20 if (typeof html5jp.graph == 'undefined') {
21     html5jp.graph = new Object();
22 }
23 
24 /* -----
25 * コンストラクタ
26 * ----- */
27 html5jp.graph.line = function (id) {
28     var elm = document.getElementById(id);
29     if (!elm) { return; }
30     if (!elm.nodeName.match(/^CANVAS$/i)) { return; }
31     if (!elm.parentNode.nodeName.match(/^DIV$/i)) { return; };
32     /* CANVAS要素 */
33     if (!elm.getContext) { return; }
34     this.canvas = elm;
35     /* 2D コンテクストの生成 */
36     this.ctx = this.canvas.getContext('2d');
37     this.canvas.style.margin = "0";
38     this.canvas.parentNode.style.position = "relative";
39     this.canvas.parentNode.style.padding = "0";
40     /* CANVAS要素の親要素となるDIV要素の幅と高さをセット */
41     this.canvas.parentNode.style.width = this.canvas.width + "px";
42     this.canvas.parentNode.style.height = this.canvas.height + "px";
43 };
44 /* -----
45 * 描画
46 * ----- */
47 html5jp.graph.line.prototype.draw = function (items, inparams) {
48     if (!this.ctx) { return; }
49     /* パラメータの初期化 */
50     var params = {
51         x: [],
52         y: [],
53         yMax: null,
54         yMin: 0,
55         backgroundColor: "#ffffff",
56         gbackgroundColor: "#ffffff",
57         gGradation: true,
58         lineWidth: 3,
59         dotRadius: 4,
60         dotType: ["diamond", "square", "disc"],
61         hLineWidth: 2,
62         hLineType: "dotted",
63         hLineColor: "#aaaaaa",
64         xAxisWidth: 2,
65         xAxisColor: "#000000",
66         yAxisWidth: 2,
67         yAxisColor: "#000000",
68         xScaleColor: "#000000",
69         xScaleFontSize: "14px",

```

```

70     xScaleFontFamily: "Arial, sans-serif",
71     yScaleColor: "#000000",
72     yScaleFontSize: "14px",
73     yScaleFontFamily: "Arial, sans-serif",
74     xCaptionColor: "#000000",
75     xCaptionFontSize: "16px",
76     xCaptionFontFamily: "Arial, sans-serif",
77     yCaptionColor: "#000000",
78     yCaptionFontSize: "16px",
79     yCaptionFontFamily: "Arial, sans-serif",
80     dLabel: true,
81     dLabelColor: "#000000",
82     dLabelFontSize: "14px",
83     dLabelFontFamily: "Arial, sans-serif",
84     legend: true,
85     legendFontSize: "16px",
86     legendFontFamily: "Arial, sans-serif",
87     legendColor: "#000000"
88 };
89 if (inparams && typeof (inparams) == 'object') {
90     for (var key in inparams) {
91         params[key] = inparams[key];
92     }
93 }
94 this.params = params;
95 /* CANVASの背景を塗る */
96 if (params.backgroundColor) {
97     this.ctx.beginPath();
98     this.ctx.fillStyle = params.backgroundColor;
99     this.ctx.fillRect(0, 0, this.canvas.width, this.canvas.height);
100 }
101 /* CANVAS要素の横幅が縦幅の1.5倍未満、または縦幅が200未満であれば凡例は強制的に非表示 */
102 if (this.canvas.width / this.canvas.height < 1.5 || this.canvas.height < 200) {
103     params.legend = false;
104 }
105 /* CANVAS要素の座標 */
106 var canvas_pos = this._getElementAbsPos(this.canvas);
107 /* 折線グラフの軸のcanvas内座標 */
108 var cpos = {
109     x0: this.canvas.width * 0.1,
110     y0: this.canvas.height * 0.9,
111     x1: this.canvas.width * 0.9,
112     y1: this.canvas.height * 0.1
113 };
114 if (typeof (params.x) == "object" && params.x.length > 0) {
115     cpos.y0 = this.canvas.height * 0.8;
116 }
117 if (typeof (params.y) == "object" && params.y.length > 0) {
118     cpos.x0 = this.canvas.width * 0.15;
119     cpos.y1 = this.canvas.height * 0.15
120 }
121 if (params.legend == true) {
122     cpos.x1 = this.canvas.width * 0.7;
123 }
124 cpos.w = cpos.x1 - cpos.x0;
125 cpos.h = cpos.y0 - cpos.y1;

```

```

126 /* 項目の数（最大10個） */
127 var item_num = items.length;
128 if (item_num > 10) { item_num = 10; }
129 /* 凡例の各種座標を算出 */
130 if (params.legend == true) {
131     /* DIV要素を仮に挿入してみて高さを調べる(1行分の高さ) */
132     var legend_tmp_s = this._getTextBoxSize('あTEST', params.legendFontSize, params.
133         legendFontFamily);
134     /* 凡例の各種座標を算出 */
135     var lpos = {
136         x: Math.round(cpos.x1 + this.canvas.width * 0.03),
137         y: Math.round((this.canvas.height - (legend_tmp_s.h * item_num + legend_tmp_s.h *
138             0.2 * (item_num - 1))) / 2),
139         h: legend_tmp_s.h
140     };
141     lpos.cx = lpos.x + Math.round(lpos.h * 2.5); /* 文字表示開始位置(x座標) */
142     lpos.cw = this.canvas.width - lpos.cx; /* 文字表示幅 */
143 }
144 /* グラフの背景を塗りつぶす */
145 if (params.gGradation == true) {
146     this.ctx.beginPath();
147     this.ctx.moveTo(cpos.x0, cpos.y0);
148     this.ctx.lineTo(cpos.x1, cpos.y0);
149     this.ctx.lineTo(cpos.x1, cpos.y1);
150     this.ctx.lineTo(cpos.x0, cpos.y1);
151     this.ctx.closePath();
152     var radgrad = this.ctx.createLinearGradient(cpos.x0, cpos.y1, cpos.x0, cpos.y0);
153     var o_gbc = this._csscolor2rgb(params.gbackgroundColor);
154     var gbc = o_gbc.r + "," + o_gbc.g + "," + o_gbc.b;
155     radgrad.addColorStop(0, "rgb(" + gbc + ")");
156     radgrad.addColorStop(1, "rgb(255,255,255)");
157     this.ctx.fillStyle = radgrad;
158     this.ctx.fill();
159 } else {
160     this.ctx.fillStyle = params.gbackgroundColor;
161     this.ctx.fillRect(cpos.x0, cpos.y1, cpos.w, cpos.h);
162 }
163 /* 全項目の最大値・最小値と項目数を算出 */
164 var max_v = null;
165 var min_v = null;
166 var max_n = 0;
167 if (params.y.length > 1) {
168     max = params.y[params.y.length - 1];
169 }
170 for (var i = 0; i < item_num; i++) {
171     var n = items[i].length;
172     if (n < 2) { continue; }
173     for (var j = 1; j < n; j++) {
174         var v = items[i][j];
175         if (isNaN(v)) {
176             throw new Error('invalid graph item data.' + n);
177         }
178         if (max_v == null) {
179             max_v = v;

```

```

180        }
181        if (min_v == null) {
182            min_v = v;
183        } else if (v <= min_v) {
184            min_v = v;
185        }
186    }
187    if (n - 1 >= max_n) {
188        max_n = n - 1;
189    }
190}
191if (typeof (params.yMin) != "number") {
192    params.yMin = 0;
193}
194if (typeof (params.yMax) != "number") {
195    params.yMax = max_v + Math.abs(max_v - min_v) * 0.1;
196}
197var v_range = Math.abs(params.yMax - params.yMin);
198/* 水平補助線 */
199if (typeof (params.hLineWidth) == "number" && params.hLineWidth > 0) {
200    var h_line_type = "dashed";
201    if (params.hLineType.match(/^\w(solid|dashed|dotted)\w$/i)) {
202        h_line_type = params.hLineType;
203    }
204    for (var i = 1; i < params.y.length; i++) {
205        var aline_x0 = cpos.x0;
206        var aline_y0 = Math.round(cpos.y0 - cpos.h * (params.y[i] - params.yMin) / v_range);
207        var aline_x1 = cpos.x1;
208        this._draw_h_aline(aline_x0, aline_y0, aline_x1, params.hLineWidth, h_line_type,
209            params.hLineColor);
210    }
211}
212/* 各項目のデフォルト色を定義 */
213var colors = ["0,128,255", "255,0,0", "191,191,191"];
214/* 折線グラフを描写 */
215var plots = new Array();
216for (var i = 0; i < item_num; i++) {
217    this.ctx.beginPath();
218    this.ctx.lineJoin = "round";
219    plots[i] = new Array();
220    var n = items[i].length;
221    for (var j = 1; j < n; j++) {
222        /* 項目の値 */
223        var v = items[i][j];
224        /* canvas座標を算出 */
225        var p = {
226            x: Math.round(cpos.x0 + cpos.w * (j - 0.5) / max_n),
227            y: Math.round(cpos.y0 - cpos.h * (v - params.yMin) / v_range),
228            v: v
229        }
230        plots[i].push(p);
231        /* 線を描画 */
232        if (j == 1) {
233            this.ctx.moveTo(p.x, p.y);
234        } else {

```

```

234         this.ctx.lineTo(p.x, p.y);
235     }
236   }
237   /* 線の太さを定義 */
238   var line_width = 1;
239   if (typeof (params.lineWidth) == "object" && !isNaN(params.lineWidth[i])) {
240     line_width = params.lineWidth[i];
241   } else if (typeof (params.lineWidth) == "number" && !isNaN(params.lineWidth)) {
242     line_width = params.lineWidth;
243   }
244   this.ctx.lineWidth = line_width;
245   /* 線の色を定義 */
246   var line_color = "rgb(" + colors[i] + ")";
247   this.ctx.strokeStyle = line_color;
248   /* 線を描画 */
249   this.ctx.stroke();
250   /* ドットの半径を特定 */
251   var dot_rad = null;
252   if (typeof (params.dotRadius) == "object" && !isNaN(params.dotRadius[i]) && params.
253     dotRadius[i] > 0) {
254     dot_rad = params.dotRadius[i];
255   } else if (typeof (params.dotRadius) == "number" && !isNaN(params.dotRadius) && params.
256     .dotRadius > 0) {
257     dot_rad = params.dotRadius;
258   }
259   /* ドットのタイプを特定 */
260   var dot_type = null;
261   if (typeof (params.dotType) == "object" && typeof (params.dotType[i]) == "string") {
262     dot_type = params.dotType[i];
263   } else if (typeof (params.dotType) == "string") {
264     dot_type = params.dotType;
265   } else {
266     dot_type = "disc";
267   }
268   /* ドットを描画 */
269   if (dot_rad > 0) {
270     for (var k = 0; k < plots[i].length; k++) {
271       this._draw_dot(plots[i][k].x, plots[i][k].y, dot_rad, dot_type, colors[i]);
272     }
273   }
274   /* データラベルを描画 */
275   if (params.dLabel == true) {
276     for (var k = 0; k < plots[i].length; k++) {
277       if (plots[i][k].x < cpos.x0 || plots[i][k].x > cpos.x1 || plots[i][k].y > cpos.
278         .y0 || plots[i][k].y < cpos.y1) {
279         continue;
280       }
281       var dlabel = plots[i][k].v.toString();
282       var margin = 1;
283       if (dot_rad != null && dot_rad > 0) {
284         margin += dot_rad;
285       }
286       var s = this._getTextBoxSize(dlabel, params.dLabelFontSize, params.
287         dLabelFontFamily);
288       var dlabel_x = plots[i][k].x - Math.round(s.w / 2);
289       var dlabel_y = plots[i][k].y - Math.round(s.h) - margin;

```

```

286         this._drawText(dlabel_x, dlabel_y, dlabel, params.dLabelFontSize, params.
287             dLabelFontFamily, params.dLabelColor);
288     }
289     /* 凡例を描画 */
290     if (params.legend == true) {
291         /* 文字 */
292         this._drawText(lpos.cx, lpos.y, items[i][0], params.legendFontSize, params.
293             legendFontFamily, params.legendColor);
294         /* 記号（野線） */
295         this._draw_h_aline(lpos.x, Math.round(lpos.y + lpos.h / 2), lpos.x + lpos.h * 2,
296             line_width, "solid", line_color);
297         /* 記号（ドット） */
298         if (dot_rad > 0) {
299             this._draw_dot(Math.round(lpos.x + lpos.h), Math.round(lpos.y + lpos.h / 2),
300                 dot_rad, dot_type, colors[i]);
301         }
302     }
303     /* グラフ描画領域外の上下を背景色で塗りつぶす */
304     this.ctx.fillStyle = params.backgroundColor;
305     this.ctx.fillRect(cpos.x0, 0, cpos.w, cpos.y1);
306     this.ctx.fillRect(cpos.x0, cpos.y0, cpos.w, this.canvas.height - cpos.y0);
307     /* x軸 */
308     if (typeof (params.xAxisWidth) == "number" && params.xAxisWidth > 0) {
309         this._draw_h_aline(cpos.x0, cpos.y0, cpos.x1, params.xAxisWidth, "solid", params.
310             xAxisColor);
311     }
312     /* y軸 */
313     if (typeof (params.yAxisWidth) == "number" && params.yAxisWidth > 0) {
314         this._draw_v_aline(cpos.x0, cpos.y0, cpos.y1, params.yAxisWidth, "solid", params.
315             yAxisColor);
316     }
317     /* x軸の目盛文字列を描画 */
318     var xscale_y_under = 0;
319     for (var i = 1; i <= max_n; i++) {
320         if (typeof (params.x[i]) != "string") { continue; }
321         if (params.x[i] == "") { continue; }
322         var s = this._getTextBoxSize(params.x[i], params.xScaleFontSize, params.
323             xScaleFontFamily);
324         var xscale_x = Math.round(cpos.x0 + cpos.w * (i - 0.5) / max_n) - Math.round(s.w / 2);
325         var xscale_y = cpos.y0 + 5;
326         this._drawText(xscale_x, xscale_y, params.x[i], params.xScaleFontSize, params.
327             xScaleFontFamily, params.xScaleColor);
328         if (xscale_y + s.h >= xscale_y_under) {
329             xscale_y_under = xscale_y + s.h;
330         }
331     }
332     /* y軸の目盛数値を描画 */
333     var yscale_y_top = this.canvas.height;
334     for (var i = 1; i < params.y.length; i++) {
335         if (typeof (params.y[i]) != "number") { continue; }
336         var v = params.y[i].toString();
337         if (v == "") { continue; }

```

```

334     if (params.y[i] < params.yMin || params.y[i] > params.yMax) { continue; }
335     var s = this._getTextBoxSize(v, params.yScaleFontSize, params.yScaleFontFamily);
336     var yscale_y = Math.round(cpos.y0 - cpos.h * (params.y[i] - params.yMin) / v_range) -
337         Math.round(s.h / 2);
338     var yscale_x = Math.round(cpos.x0 - s.w) - 5;
339     this._drawText(yscale_x, yscale_y, v, params.yScaleFontSize, params.yScaleFontFamily,
340     params.yScaleColor);
341     if (yscale_y <= yscale_y_top) {
342         yscale_y_top = yscale_y;
343     }
344     /* x軸のキャプションを描画 */
345     if (typeof (params.x[0]) == "string" && params.x[0] != "") {
346         var s = this._getTextBoxSize(params.x[0], params.xCaptionFontSize, params.
347             xCaptionFontFamily);
348         var xcaption_y = cpos.y0 + 5;
349         if (xscale_y_under > 0) {
350             xcaption_y = xscale_y_under + 5;
351         }
352         var xcaption_x = Math.round(cpos.x0 + (cpos.w / 2) - (s.w / 2));
353         this._drawText(xcaption_x, xcaption_y, params.x[0], params.xCaptionFontSize, params.
354             xCaptionFontFamily, params.xCaptionColor);
355     }
356     /* y軸のキャプションを描画 */
357     if (typeof (params.y[0]) == "string" && params.y[0] != "") {
358         var s = this._getTextBoxSize(params.y[0], params.yCaptionFontSize, params.
359             yCaptionFontFamily);
360         var ycaption_y = yscale_y_top - s.h - 5;
361         if (yscale_y_top > cpos.y1) {
362             ycaption_y = cpos.y1 - s.h - 5;
363         }
364         var ycaption_x = Math.round(cpos.x0 - (s.w / 2));
365         if (ycaption_x < 5) {
366             ycaption_x = 5;
367         }
368         this._drawText(ycaption_x, ycaption_y, params.y[0], params.yCaptionFontSize, params.
369             yCaptionFontFamily, params.yCaptionColor);
370     }
371 }
372 /* -----
373 * 以下、内部関数
374 * -----
375 * ----- */
376 html5jp.graph.line.prototype._draw_v_aline = function (x0, y0, y1, width, type, color) {
377     color = this._csscolor2rgb(color);
378     this.ctx.beginPath();
379     color = "rgb(" + color.r + "," + color.g + "," + color.b + ")"
380     this.ctx.strokeStyle = color;
381     this.ctx.lineWidth = width;
382     if (type == "solid") {
383         this.ctx.moveTo(x0, y0);
384         this.ctx.lineTo(x0, y1);

```

```

384     this.ctx.stroke();
385 } else if (type == "dashed" || (type == "dotted" && width < 2)) {
386     var y = y0;
387     while (1) {
388         if (y - width * 4 < y1) { break; }
389         this.ctx.moveTo(x0, y);
390         y = y - width * 4;
391         this.ctx.lineTo(x0, y);
392         this.ctx.stroke();
393         if (y - width * 2 < y1) { break; }
394         y = y - width * 2;
395     }
396 } else if (type == "dotted") {
397     this.ctx.fillStyle = color;
398     var y = y0;
399     while (1) {
400         if (y - width * 2 < y1) { break; }
401         this.ctx.arc(x0, y, width / 2, 0, Math.PI * 2, false);
402         this.ctx.fill();
403         if (y - width * 2 < y1) { break; }
404         y = y - width * 2;
405     }
406 }
407 };
408 /* ----- */
409 水平補助線を描画
410 * -----
411 html5jp.graph.line.prototype._draw_h_aeline = function (x0, y0, x1, width, type, color) {
412     color = this._csscolor2rgb(color);
413     this.ctx.beginPath();
414     color = "rgb(" + color.r + "," + color.g + "," + color.b + ")";
415     this.ctx.strokeStyle = color;
416     this.ctx.lineWidth = width;
417     if (type == "solid") {
418         this.ctx.moveTo(x0, y0);
419         this.ctx.lineTo(x1, y0);
420         this.ctx.stroke();
421     } else if (type == "dashed" || (type == "dotted" && width < 2)) {
422         var x = x0;
423         while (1) {
424             if (x + width * 4 > x1) { break; }
425             this.ctx.moveTo(x, y0);
426             x = x + width * 4;
427             this.ctx.lineTo(x, y0);
428             this.ctx.stroke();
429             if (x + width * 2 > x1) { break; }
430             x = x + width * 2;
431         }
432     } else if (type == "dotted") {
433         this.ctx.fillStyle = color;
434         var x = x0;
435         while (1) {
436             if (x + width * 2 > x1) { break; }
437             this.ctx.arc(x, y0, width / 2, 0, Math.PI * 2, false);
438             this.ctx.fill();
439             if (x + width * 2 > x1) { break; }

```

```

440         x = x + width * 2;
441     }
442 }
443 */
444 /* -----
445 折線のドットを描画
446 * -----
447 html5jp.graph.line.prototype._draw_dot = function (x, y, rad, type, color) {
448     this.ctx.beginPath();
449     this.ctx.fillStyle = "rgb(" + color + ")";
450     if (type == "disc") {
451         this.ctx.arc(x, y, rad, 0, Math.PI * 2, false);
452     } else if (type == "square") {
453         this.ctx.moveTo(x - rad, y - rad);
454         this.ctx.lineTo(x + rad, y - rad);
455         this.ctx.lineTo(x + rad, y + rad);
456         this.ctx.lineTo(x - rad, y + rad);
457     } else if (type == "triangle") {
458         this.ctx.moveTo(x, y - rad);
459         this.ctx.lineTo(x + rad, y + rad);
460         this.ctx.lineTo(x - rad, y + rad);
461     } else if (type == "i-triangle") {
462         this.ctx.moveTo(x, y + rad);
463         this.ctx.lineTo(x + rad, y - rad);
464         this.ctx.lineTo(x - rad, y - rad);
465     } else if (type == "diamond") {
466         this.ctx.moveTo(x, y - rad);
467         this.ctx.lineTo(x + rad, y);
468         this.ctx.lineTo(x, y + rad);
469         this.ctx.lineTo(x - rad, y);
470     } else {
471         this.ctx.arc(x, y, rad, 0, Math.PI * 2, false);
472     }
473     this.ctx.closePath();
474     this.ctx.fill();
475 };
476
477 /* -----
478 文字列を描画
479 * -----
480 html5jp.graph.line.prototype._drawText = function (x, y, text, font_size, font_family, color)
481 {
482     var div = document.createElement('DIV');
483     div.appendChild(document.createTextNode(text));
484     div.style.fontSize = font_size;
485     div.style.fontFamily = font_family;
486     div.style.color = color;
487     div.style.margin = "0";
488     div.style.padding = "0";
489     div.style.position = "absolute";
490     div.style.left = x.toString() + "px";
491     div.style.top = y.toString() + "px";
492     this.canvas.parentNode.appendChild(div);
493 }
494 /* -----
495 文字列表示領域のサイズを取得

```

```

495 * -----
496 html5jp.graph.line.prototype._getTextBoxSize = function (text, font_size, font_family) {
497     var tmpdiv = document.createElement('DIV');
498     tmpdiv.appendChild(document.createTextNode(text));
499     tmpdiv.style.fontSize = font_size;
500     tmpdiv.style.fontFamily = font_family;
501     tmpdiv.style.margin = "0";
502     tmpdiv.style.padding = "0";
503     tmpdiv.style.visible = "hidden";
504     tmpdiv.style.position = "absolute";
505     tmpdiv.style.left = "0px";
506     tmpdiv.style.top = "0px";
507     this.canvas.parentNode.appendChild(tmpdiv);
508     var o = {
509         w: tmpdiv.offsetWidth,
510         h: tmpdiv.offsetHeight
511     };
512     tmpdiv.parentNode.removeChild(tmpdiv);
513     return o;
514 }
515
516 /* -----
517 ブラウザ表示領域左上端を基点とする座標系におけるelmの左上端の座標
518 * -----
519 html5jp.graph.line.prototype._getElementAbsPos = function (elm) {
520     var obj = new Object();
521     obj.x = elm.offsetLeft;
522     obj.y = elm.offsetTop;
523     while (elm.offsetParent) {
524         elm = elm.offsetParent;
525         obj.x += elm.offsetLeft;
526         obj.y += elm.offsetTop;
527     }
528     return obj;
529 }
530
531 /* -----
532 * CSS色文字列をRGBに変換
533 * -----
534 html5jp.graph.line.prototype._csscolor2rgb = function (c) {
535     if (!c) { return null; }
536     var color_map = {
537         black: "#000000",
538         gray: "#808080",
539         silver: "#c0c0c0",
540         white: "#ffffff",
541         maroon: "#800000",
542         red: "#ff0000",
543         purple: "#800080",
544         fuchsia: "#ff00ff",
545         green: "#008000",
546         lime: "#00FF00",
547         olive: "#808000",
548         yellow: "#FFFF00",
549         navy: "#000080",
550         blue: "#0000ff",

```

```

551     teal: "#008080",
552     aqua: "#00FFFF"
553   };
554   c = c.toLowerCase();
555   var o = new Object();
556   if (c.match(/^[a-zA-Z]+$/)) && color_map[c] ) {
557     c = color_map[c];
558   }
559   var m = null;
560   if (m = c.match(/^#\([a-f\d]{2}\)([a-f\d]{2})([a-f\d]{2})$/i)) {
561     o.r = parseInt(m[1], 16);
562     o.g = parseInt(m[2], 16);
563     o.b = parseInt(m[3], 16);
564   } else if (m = c.match(/^#[[a-f\d]{1})([a-f\d]{1})([a-f\d]{1})$/i)) {
565     o.r = parseInt(m[1] + "0", 16);
566     o.g = parseInt(m[2] + "0", 16);
567     o.b = parseInt(m[3] + "0", 16);
568   } else if (m = c.match(/^rgba*\((\s*(\d+),\s*(\d+),\s*(\d+)/i)) {
569     o.r = m[1];
570     o.g = m[2];
571     o.b = m[3];
572   } else if (m = c.match(/^rgba*\((\s*(\d+)\%,\s*(\d+)\%,\s*(\d+)\%)/i)) {
573     o.r = Math.round(m[1] * 255 / 100);
574     o.g = Math.round(m[2] * 255 / 100);
575     o.b = Math.round(m[3] * 255 / 100);
576   } else {
577     return null;
578   }
579   return o;
580 }

```

src. B-4: 円グラフ HTML 改良版

```

1  <!DOCTYPE html>
2  <html lang="ja">
3
4  <head>
5    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
6    <script type="text/javascript" src=".no2_circle.js"></script>
7    <script type="text/javascript">
8      window.onload = function () {
9        var cg = new html5jp.graph.circle("graph02");
10       if (!cg) {
11         return;
12       }
13       var items = [
14         ["商品a", 100],
15         ["商品b", 60],
16         ["商品c", 25],
17         ["商品d", 10]
18       ];
19       var params = {
20         captionNum: false,
21       };
22       cg.draw(items, params);
23     };

```

```

24  </script>
25  <title>円グラフ</title>
26 </head>
27
28 <body>
29
30  <div><canvas width="600" height="450" id="graph02"></canvas></div>
31
32 </body>
33
34 </html>

```

src. B-5: 円グラフ JavaScript 改良版

```

1 // Copyright 2007-2009 futomi http://www.html5.jp/
2 //
3 // Licensed under the Apache License, Version 2.0 (the "License");
4 // you may not use this file except in compliance with the License.
5 // You may obtain a copy of the License at
6 //
7 //   http://www.apache.org/licenses/LICENSE-2.0
8 //
9 // Unless required by applicable law or agreed to in writing, software
10 // distributed under the License is distributed on an "AS IS" BASIS,
11 // WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 // See the License for the specific language governing permissions and
13 // limitations under the License.
14 //
15 // circle.js v1.0.2
16
17 if (typeof html5jp == 'undefined') {
18     html5jp = new Object();
19 }
20 if (typeof html5jp.graph == 'undefined') {
21     html5jp.graph = new Object();
22 }
23
24 /* -----
25 * コンストラクタ
26 * ----- */
27 html5jp.graph.circle = function (id) {
28     var elm = document.getElementById(id);
29     if (!elm) { return; }
30     if (!elm.nodeName.match(/^CANVAS$/i)) { return; }
31     if (!elm.parentNode.nodeName.match(/^DIV$/i)) { return; };
32     /* CANVAS要素 */
33     if (!elm.getContext) { return; }
34     this.canvas = elm;
35     /* 2D コンテクストの生成 */
36     this.ctx = this.canvas.getContext('2d');
37     this.canvas.style.margin = "0";
38     this.canvas.parentNode.style.position = "relative";
39     this.canvas.parentNode.style.padding = "0";
40     /* CANVAS要素の親要素となるDIV要素の幅と高さをセット */
41     this.canvas.parentNode.style.width = this.canvas.width + "px";
42     this.canvas.parentNode.style.height = this.canvas.height + "px";

```

```

43 };
44 /* -----
45 * 描画
46 * ----- */
47 html5jp.graph.circle.prototype.draw = function (items, inparams) {
48     if (!this.ctx) { return; }
49     /* パラメータの初期化 */
50     var params = {
51         backgroundColor: null,
52         shadow: true,
53         border: false,
54         caption: false,
55         captionNum: true,
56         captionRate: true,
57         fontSize: "16px",
58         fontFamily: "Arial,sans-serif",
59         textShadow: true,
60         captionColor: "#ffffff",
61         startAngle: -90,
62         legend: true,
63         legendFontSize: "16px",
64         legendFontFamily: "Arial,sans-serif",
65         legendColor: "#000000",
66         otherCaption: "other"
67     };
68     if (inparams && typeof (inparams) == 'object') {
69         for (var key in inparams) {
70             params[key] = inparams[key];
71         }
72     }
73     this.params = params;
74     /* CANVASの背景を塗る */
75     if (params.backgroundColor) {
76         this.ctx.beginPath();
77         this.ctx.fillStyle = params.backgroundColor;
78         this.ctx.fillRect(0, 0, this.canvas.width, this.canvas.height);
79     }
80     /* CANVAS要素の横幅が縦幅の1.5倍未満、または縦幅が200未満であれば凡例は強制的に非表示 */
81     if (this.canvas.width / this.canvas.height < 1.5 || this.canvas.height < 200) {
82         params.legend = false;
83     }
84     /* CANVAS要素の座標 */
85     var canvas_pos = this._getElementAbsPos(this.canvas);
86     /* 円グラフの中心座標と半径 */
87     var cpos = {
88         x: this.canvas.width / 2,
89         y: this.canvas.height / 2,
90         r: Math.min(this.canvas.width, this.canvas.height) * 0.8 / 2
91     };
92     /* 10項目を超えていれば、10項目目以降を"その他"に統合 */
93     items = this._fold_items(items);
94     var item_num = items.length;
95     /* 凡例表示の場合の円グラフ中心x座標と、凡例の各種座標を算出 */
96     if (params.legend == true) {
97         /* 円グラフ中心x座標 */
98         cpos.x = this.canvas.height * 0.1 + cpos.r;

```

```

99  /* DIV要素を仮に挿入してみて高さを調べる(1行分の高さ) */
100 var tmpdiv = document.createElement('DIV');
101 tmpdiv.appendChild(document.createTextNode('あTEST'));
102 tmpdiv.style.fontSize = params.legendFontSize;
103 tmpdiv.style.fontFamily = params.legendFontFamily;
104 tmpdiv.style.color = params.legendColor;
105 tmpdiv.style.margin = "0";
106 tmpdiv.style.padding = "0";
107 tmpdiv.style.visible = "hidden";
108 tmpdiv.style.position = "absolute";
109 tmpdiv.style.left = canvas_pos.x.toString() + "px";
110 tmpdiv.style.top = canvas_pos.y.toString() + "px";
111 this.canvas.parentNode.appendChild(tmpdiv);
112 /* 凡例の各種座標を算出 */
113 var lpos = {
114     x: this.canvas.height * 0.2 + cpos.r * 2,
115     y: (this.canvas.height - (tmpdiv.offsetHeight * item_num + tmpdiv.offsetHeight *
116         0.2 * (item_num - 1))) / 2,
117     h: tmpdiv.offsetHeight
118 };
119 lpos.cx = lpos.x + lpos.h * 1.4; // 文字表示開始位置(x座標)
120 lpos.cw = this.canvas.width - lpos.cx; // 文字表示幅
121 /* 仮に挿入したDIV要素を削除 */
122 tmpdiv.parentNode.removeChild(tmpdiv);
123 }
124 /* 外円の影 */
125 if (params.shadow == true) {
126     this._make_shadow(cpos);
127 }
128 /* 外円を黒で塗りつぶす */
129 this.ctx.beginPath();
130 this.ctx.arc(cpos.x, cpos.y, cpos.r, 0, Math.PI * 2, false)
131 this.ctx.closePath();
132 this.ctx.fillStyle = "black";
133 this.ctx.fill();
134 /* 全項目の値の合計を算出 */
135 var sum = 0;
136 for (var i = 0; i < item_num; i++) {
137     var n = items[i][1];
138     if (isNaN(n) || n < 0) {
139         throw new Error('invalid graph item data.' + n);
140     }
141     sum += n;
142 }
143 if (sum <= 0) {
144     throw new Error('invalid graph item data.');
145 }
146 /* 各項目のデフォルト色を定義 */
147 var colors = ["255,0,0", "65,105,255", "255,255,0", "0,255,255"];
148 /* 円グラフを描写 */
149 var rates = new Array();
150 var startAngle = this._degree2radian(params.startAngle);
151 for (var i = 0; i < item_num; i++) {
152     /* 項目の名前 */
153     var cap = items[i][0];
154     /* 項目の値 */

```

```

154 var n = items[i][1];
155 /* 比率 */
156 var r = n / sum;
157 /* パーセント */
158 var p = Math.round(r * 1000) / 10;
159 /* 描写角度（ラジアン） */
160 var rad = this._degree2radian(360 * r);
161 /* 円弧終点角度 */
162 endAngle = startAngle + rad;
163 /* 円グラフの色を特定 */
164 var rgb = colors[i];
165 var rgbo = this._csscolor2rgb(items[i][2]);
166 if (rgbo) {
167     rgb = rgbo.r + "," + rgbo.g + "," + rgbo.b;
168 }
169 /* */
170 if (n > 0) {
171     /* 円弧を描画 */
172     this.ctx.beginPath();
173     if (p >= 100) { // for excanvas
174         this.ctx.arc(cpos.x, cpos.y, cpos.r, 0, Math.PI * 2, false);
175     } else {
176         this.ctx.moveTo(cpos.x, cpos.y);
177         this.ctx.lineTo(
178             cpos.x + cpos.r * Math.cos(startAngle),
179             cpos.y + cpos.r * Math.sin(startAngle)
180         );
181         this.ctx.arc(cpos.x, cpos.y, cpos.r, startAngle, endAngle, false);
182     }
183     this.ctx.closePath();
184     /* 円グラフのグラデーションをセット */
185     var radgrad = this.ctx.createRadialGradient(cpos.x, cpos.y, 0, cpos.x, cpos.y,
186         cpos.r);
187     radgrad.addColorStop(0, "rgba(" + rgb + ",1)");
188     radgrad.addColorStop(0.7, "rgba(" + rgb + ",0.85)");
189     radgrad.addColorStop(1, "rgba(" + rgb + ",0.75)");
190     this.ctx.fillStyle = radgrad;
191     this.ctx.fill();
192     /* 円弧の枠線 */
193     if (params.border == true) {
194         this.ctx.stroke();
195     }
196     /* キャプション */
197     var drate;
198     var fontSize;
199     if (r <= 0.03) {
200         drate = 1.1;
201     } else if (r <= 0.05) {
202         drate = 0.9;
203     } else if (r <= 0.1) {
204         drate = 0.8;
205     } else if (r <= 0.15) {
206         drate = 0.7;
207     } else {
208         drate = 0.6;
209     }

```

```

209     var cp = {
210         x: cpos.x + (cpos.r * drate) * Math.cos((startAngle + endAngle) / 2),
211         y: cpos.y + (cpos.r * drate) * Math.sin((startAngle + endAngle) / 2)
212     };
213     var div = document.createElement('DIV');
214     if (r <= 0.05) {
215         if (params.captionRate == true) {
216             div.appendChild(document.createTextNode(p + "%"));
217         }
218     } else {
219         if (params.caption == true) {
220             div.appendChild(document.createTextNode(cap));
221             if (params.captionNum == true || params.captionRate == true) {
222                 div.appendChild(document.createElement('BR'));
223             }
224         }
225         if (params.captionRate == true) {
226             div.appendChild(document.createTextNode(p + "%"));
227         }
228         if (params.captionNum == true) {
229             var numCap = n;
230             if (params.caption == true || params.captionRate == true) {
231                 numCap = "(" + numCap + ")";
232             }
233             div.appendChild(document.createTextNode(numCap));
234         }
235     }
236     div.style.position = 'absolute';
237     div.style.textAlign = 'center';
238     div.style.color = params.captionColor;
239     div.style.fontSize = params.fontSize;
240     div.style.fontFamily = params.fontFamily;
241     div.style.margin = "0";
242     div.style.padding = "0";
243     div.style.visible = "hidden";
244     if (params.textShadow == true) {
245         var dif = [[0, 1], [0, -1], [1, 0], [1, 1], [1, -1], [-1, 0], [-1, 1], [-1,
246             -1]];
247         for (var j = 0; j < 8; j++) {
248             var s = div.cloneNode(true);
249             this.canvas.parentNode.appendChild(s);
250             s.style.color = "black";
251             s.style.left = (parseFloat(cp.x - s.offsetWidth / 2 + dif[j][0])).toString()
252                 () + "px";
253             s.style.top = (cp.y - s.offsetHeight / 2 + dif[j][1]).toString() + "px";
254         }
255     }
256     this.canvas.parentNode.appendChild(div);
257     div.style.left = (cp.x - div.offsetWidth / 2).toString() + "px";
258     div.style.top = (cp.y - div.offsetHeight / 2).toString() + "px";
259 }
/* 凡例 */
if (params.legend == true) {
    /* 文字 */
    var ltxt = document.createElement('DIV');
    ltxt.appendChild(document.createTextNode(cap));

```

```

263     ltxt.style.position = "absolute";
264     ltxt.style.fontSize = params.legendFontSize;
265     ltxt.style.fontFamily = params.legendFontFamily;
266     ltxt.style.color = params.legendColor;
267     ltxt.style.margin = "0";
268     ltxt.style.padding = "0";
269     ltxt.style.left = lpos.cx.toString() + "px";
270     ltxt.style.top = lpos.y.toString() + "px";
271     ltxt.style.width = (lpos.cw).toString() + "px";
272     ltxt.style.whiteSpace = "nowrap";
273     ltxt.style.overflow = "hidden";
274     this.canvas.parentNode.appendChild(ltxt);
275     /* 記号の影 */
276     if (params.shadow == true) {
277         this.ctx.beginPath();
278         this.ctx.rect(lpos.x + 1, lpos.y + 1, lpos.h, lpos.h);
279         this.ctx.fillStyle = "#222222";
280         this.ctx.fill();
281     }
282     /* 記号 */
283     this.ctx.beginPath();
284     this.ctx.rect(lpos.x, lpos.y, lpos.h, lpos.h);
285     this.ctx.fillStyle = "black";
286     this.ctx.fill();
287     this.ctx.beginPath();
288     this.ctx.rect(lpos.x, lpos.y, lpos.h, lpos.h);
289     var symbolr = {
290         x: lpos.x + lpos.h / 2,
291         y: lpos.y + lpos.h / 2,
292         r: Math.sqrt(lpos.h * lpos.h * 2) / 2
293     };
294     var legend_radgrad = this.ctx.createRadialGradient(symbolr.x, symbolr.y, 0,
295         symbolr.x, symbolr.y, symbolr.r);
296     legend_radgrad.addColorStop(0, "rgba(" + rgb + ",1)");
297     legend_radgrad.addColorStop(0.7, "rgba(" + rgb + ",0.85)");
298     legend_radgrad.addColorStop(1, "rgba(" + rgb + ",0.75)");
299     this.ctx.fillStyle = legend_radgrad;
300     this.ctx.fill();
301     /* */
302     lpos.y = lpos.y + lpos.h * 1.2;
303 }
304 /* */
305 startAngle = endAngle;
306 }
307
308 /* -----
309 * 以下、内部関数
310 * -----
311 */
312 /* -----
313 項目を10項目以内にまとめる
314 * -----
315 html5jp.graph.circle.prototype._fold_items = function (items) {
316     var len = items.length;
317     if (len <= 10) { return items; }

```

```

318     var n = 0;
319     for (var i = 9; i < len; i++) {
320         n += items[i][1];
321     }
322     var newitems = items.slice(0, 10);
323     newitems[9] = [this.params.otherCaption, n];
324     return newitems;
325 };
326
327 /* -----
328 ブラウザ表示領域左上端を基点とする座標系におけるelmの左上端の座標
329 * ----- */
330 html5jp.graph.circle.prototype._getElementAbsPos = function (elm) {
331     var obj = new Object();
332     obj.x = elm.offsetLeft;
333     obj.y = elm.offsetTop;
334     while (elm.offsetParent) {
335         elm = elm.offsetParent;
336         obj.x += elm.offsetLeft;
337         obj.y += elm.offsetTop;
338     }
339     return obj;
340 };
341
342 /* -----
343 * シャドーの生成
344 * ----- */
345 html5jp.graph.circle.prototype._make_shadow = function (cpos) {
346     this.ctx.beginPath();
347     this.ctx.arc(cpos.x + 5, cpos.y + 5, cpos.r, 0, Math.PI * 2, false)
348     this.ctx.closePath();
349     var radgrad = this.ctx.createRadialGradient(cpos.x + 5, cpos.y + 5, 0, cpos.x + 5, cpos.y
350         + 5, cpos.r);
351     if (document.uniqueID) {
352         radgrad.addColorStop(0, '#555555');
353     } else {
354         radgrad.addColorStop(0, 'rgba(0,0,0,1)');
355         radgrad.addColorStop(0.93, 'rgba(0,0,0,1)');
356         radgrad.addColorStop(1, 'rgba(0,0,0,0)');
357     }
358     this.ctx.fillStyle = radgrad;
359     this.ctx.fill();
360 };
361
362 /* -----
363 * 角度の度をラジアンに変換
364 * ----- */
365 html5jp.graph.circle.prototype._degree2radian = function (degree) {
366     return (Math.PI / 180) * degree;
367 };
368
369 /* -----
370 * CSS色文字列をRGBに変換
371 * ----- */
372 html5jp.graph.circle.prototype._csscolor2rgb = function (c) {
373     if (!c) { return null; }

```

```

373  var color_map = {
374    black: "#000000",
375    gray: "#808080",
376    silver: "#c0c0c0",
377    white: "#ffffff",
378    maroon: "#800000",
379    red: "#ff0000",
380    purple: "#800080",
381    fuchsia: "#ff00ff",
382    green: "#008000",
383    lime: "#00ff00",
384    olive: "#808000",
385    yellow: "#ffff00",
386    navy: "#000080",
387    blue: "#0000ff",
388    teal: "#008080",
389    aqua: "#00ffff"
390  };
391  c = c.toLowerCase();
392  var o = new Object();
393  if (c.match(/^[a-zA-Z]+$/)) && color_map[c]) {
394    c = color_map[c];
395  }
396  var m = null;
397  if (m = c.match(/^#[[a-f\d]{2}]([a-f\d]{2})([a-f\d]{2})$/i)) {
398    o.r = parseInt(m[1], 16);
399    o.g = parseInt(m[2], 16);
400    o.b = parseInt(m[3], 16);
401  } else if (m = c.match(/^#[[a-f\d]{1}]([a-f\d]{1})([a-f\d]{1})$/i)) {
402    o.r = parseInt(m[1] + "0", 16);
403    o.g = parseInt(m[2] + "0", 16);
404    o.b = parseInt(m[3] + "0", 16);
405  } else if (m = c.match(/^rgba*\(\s*(\d+),\s*(\d+),\s*(\d+)/i)) {
406    o.r = m[1];
407    o.g = m[2];
408    o.b = m[3];
409  } else if (m = c.match(/^rgba*\(\s*(\d+)\%,\s*(\d+)\%,\s*(\d+)\%$/i)) {
410    o.r = Math.round(m[1] * 255 / 100);
411    o.g = Math.round(m[2] * 255 / 100);
412    o.b = Math.round(m[3] * 255 / 100);
413  } else {
414    return null;
415  }
416  return o;
417};

```