

1 日目 HTML

1. はじめの一步
2. HTML とは
3. 文の強調と区切り
4. リンクと画像
5. 練習問題 1
6. 文字のスタイル
7. 領域のスタイル
8. セレクタとスタイルシート
9. 練習問題 2
10. リスト
11. 表
12. 練習問題 3

1-1 はじめの一步

エディタとブラウザの使い方を覚えよう。

ポイント

- ・エディタとは文章やプログラムを書くためのソフトである。この授業では「TeraPad」を使う。
- ・ブラウザとはインターネット上にある web ページを見るためのソフトである。
この授業では「Google Chrome」を使う。

課題

1. デスクトップに自分の名前のフォルダを作ろう。これからの課題は全てこの中で行う。
2. TeraPad を起動しよう。
3. 次のように書く。空白に見える部分はスペースではなくて Tab（左上の方のキー）で打つ。

```
1 | <html>↓
2 |   <body>↓
3 |     こんにちは！↓
4 |   </body>↓
5 | </html>↓
6 | [EOF]
```

4. 「1_1_intro.html」という名前で、さっき作ったフォルダの中に保存する。
5. ダブルクリックして Google Chrome で開いてみる。
6. こんにちは！と表示されるはず。
7. TeraPad に戻って、「すべて選択」をして、BackSpace キーを押そう。
8. 全て消えてしまった！こういう時は慌てずに「元に戻す」をしよう。
9. TeraPad に戻って、「コピー」と「貼り付け」を使って次のように書き換えよう。

```
1 | <html>↓
2 |   <body>↓
3 |     こんにちは！↓
4 |     こんにちは！↓
5 |     こんにちは！↓
6 |     こんにちは！↓
7 |     こんにちは！↓
8 |   </body>↓
9 | </html>↓
10| [EOF]
```

10. 上書き保存して、またブラウザで開いてみる。こんにちは！が5つ横に並んで表示されるはず。
11. 右クリックして、「要素の検証」を押してみよう。
12. 三角形のボタンを押すと開いたり閉じたり出来る。ほかにもいろいろ押してみよう。

1-2 HTML とは

ポイント

- ・ HTML は web ページの設計図である。
- ・ HTML は入れ子になった「タグ」からなる。
- ・ タグは基本的に <タグ名> 中身 </タグ名> という構造をしている。
- ・ タグは、囲った中身の「意味」をブラウザに教える。
 - <html> : 中身が HTML 文書であることを表す。ブラウザはこの中身を表示する。
 - <head> : web ページの本文ではなく、ページ全体の情報であることを示す。
 - <title> : web ページのタイトルを表す。<head>の中身には書かないといけない。
 - <body> : 本文であることを表す。
 - <div> : 本文の中の、一つのかたまりを表す。
 - <script> : プログラムを表す。これは明日以降扱う。
- ・ タグの中にタグがあってもいい <html> <body> </body> </html>
- ・ しかしこれはいけない <html> <body> </html> </body>

課題

- ・ 試しに wikipedia の HTML のページのソース（設計図のこと）を見てみよう。

<http://ja.wikipedia.org/wiki/HTML>

にアクセスして、どこかを右クリックして「要素の検証」を押してみよう。

これで、wikipedia のページがどういう設計図から書かれているのかが分かる。上に書いたタグの説明を見ながら、ながめてみよう。HTML 文の上にカーソルを合わせると、その文がどの部分を書いているのかが上の画面に表示される。

- ・ 次の HTML を書こう。ファイル名：1_2_html.html

```
1 | <html>↓
2 |   <head>↓
3 |     <title>HTMLとは</title>↓
4 |   </head>↓
5 |   <body>↓
6 |     <div>HTMLはwebページの設計図。</div>↓
7 |     <div>タグで文章や他のタグを囲うことで、ブラウザにその意味を伝える。</div>↓
8 |   </body>↓
9 | </html>↓
10| [EOF]
```

書けたら次のことを確認しよう。

「HTML とは」、というタイトルはどこに表示されているか？

要素の検証で<div>タグで囲われた範囲にマウスを合わせると、どこの色が変わるか？

1-3 文の強調と区切り

ポイント

- ・文章を読みやすくするために、段落や見出し、強調をするためのタグがある。
 - <p>：囲んだ部分が一つの段落として扱われる。
 - <h1>, <h2>, …, <h6>：囲んだ部分が見出しとなる。数字が小さいほど大きい。
 - ：囲んだ部分を太字にする。
 - <i>：囲んだ部分を斜体にする。
- ・改行や区切り線には次のタグを使う。これらのタグは、囲む部分がないため相方がいらぬ。
つまり、
だけ書けば改行され、</br>は必要ない。
 -
：改行
 - <hr>：区切り線

課題

- ・上で出てきたタグを使って、次のように表示されるような HTML を書こう。
ファイル名：1_3_kyouchou.html
1-2 で作った HTML をコピーしてくると速く書ける。

検索エンジンとHTML

webページは人間だけでなく、Googleなどの検索サイトにあるコンピュータが読むこともある。
タグを使って文章を強調するのは、日本語の分からないコンピュータに文の中の大事なところを伝えるという意味もある。

「検索エンジンとHTML」と書かれている部分は<h3>により強調されている。本文の二文は同じ<p>の中にあり、
で改行されている。Googleは斜体、コンピュータは太字である。
これらすべての要素は同じ<div>の中にある。

(解答 自信のある人は見ないでやってみよう)

```
1 | <html>↓
2 |   <head>↓
3 |     <title>文の区切りと強調</title>↓
4 |   </head>↓
5 |   <body>↓
6 |     <div>↓
7 |       <h3>検索エンジンとHTML</h3>↓
8 |       <p>↓
9 |         webページは人間だけでなく、<i>Google</i>などの↓
10 |         検索サイトにあるコンピュータが読むこともある。<br>↓
11 |         タグを使って文章を強調するのは、↓
12 |         日本語の分からない<b>コンピュータに</b>文の中の↓
13 |         大事なところを伝えるという意味もある。↓
14 |       <hr>↓
15 |     </p>↓
16 |   </div>↓
17 </body>↓
18 </html>↓
19 [EOF]
```

1-4 リンクと画像

ポイント

- ・ web ページは一つの html ファイルだけで作られているわけではない。
- ・ 別の html ファイルに飛ばしたいときは、「リンク」を使う。
` ... `
- ・ リンク先は次のように書く。
リンク元と同じフォルダの中にあるとき：target.html
同じフォルダにある「folder」というフォルダの中にあるとき：./folder/target.html
インターネット上にある別のサイトに飛ばすとき：http://google.co.jp
- ・ web ページの中に絵や図を入れたいときは次のように書く。
``
画像ファイル名のところは、リンク先と同じように書く。

課題

- ・ まずは今まで作った 3 つの HTML ファイルへのリンクを作ってみよう。

ファイル名：1_4_link.html

```
1 <html>↓
2 <head>↓
3 <title>リンクと画像</title>↓
4 </head>↓
5 <body>↓
6 <div>↓
7 <a href="1_1_intro.html">はじめの一步</a><br>↓
8 <a href="1_2_html.html">HTMLとは</a><br>↓
9 <a href="1_3_kyouchou.html">文の強調と区切り</a><br>↓
10 </div>↓
11 </body>↓
12 </html>↓
13 [EOF]
```

<a>タグで囲った部分がリンクとして機能していることがわかるだろう。

- ・ 次に Google へのリンクを追加してみよう。(http://google.co.jp)
- ・ 「ペイント」を起動して適当な画像を作成しよう。
ペイントはスタートメニュー→プログラム→アクセサリの中にある。
使い方がわからない人は手を上げてくれれば教えます。
- ・ ファイル名を「img1_4.png」として HTML と同じフォルダに保存しよう。
- ・ HTML に

``

の一文を追加して、作った画像が表示されるか確かめてみよう。

1-5 練習問題

リンクの<a>タグの中身としてタグを埋め込むと、画像をリンクとして使うことが出来る。

```
<a href="リンク先"></a>
```

この機能を使って、次のようなページを作rinaさい。



画像をクリックすると、その画像の検索サイトに飛ぶ。

Google, Yahoo, bing はいずれも検索サイトの名前であり、アドレスは検索すればすぐに見つかる。

画像は次の手順で手に入れる。ちなみに bing は windows を作っている microsoft 社の検索エンジン。

1. 検索サイトに飛んで、PrintScreen (PrtScn) キーを押す。
2. ペイントで「編集」メニューから「貼り付け」をする。
3. ペイントでロゴ画像の部分だけ「選択」して、「コピー」する。
4. 描画エリアのサイズを小さくして（描画エリアの右下にある小さな四角形を引っ張る）、ロゴが入る大きさまで小さくしたら「貼り付け」を行う。
5. 名前をつけて保存

できたらひと休み(^^) _旦~~

1-6 文字のスタイル

ポイント

- ・タグで囲われた範囲に追加の意味をもたせるときに、「属性」を使う。

<タグ名 属性名="属性の内容"> … </タグ名>

リンクの所で使った href や src は属性の 1 つだった。

- ・文字色や背景色、位置の指定には、「style 属性」を使う。
- ・ここでは style 属性の項目として次のものを使ってみよう。

- 1.style="color:色;" … **文字色**を「色」にする。

色の指定方法は、色名、16 進数表記、RGB 表記の 3 通りがある。

たとえば赤なら red, #FF0000, 255 0 0 のどれかを書く。

次のサイトを参考にすると良い。

<http://www.htmq.com/color/colortname.shtml>

- 2.style="background-color:色;" … **背景色**を指定する

- 3.style="border:形 太さ 色;" … **枠線**の形, 太さ, 色を指定する。

形: solid … 直線, double … 二重線

太さ: 1px, 2px, 3px … のように太さ+px で指定する。

Px はピクセルと読み、画面の光る点 1 マスのこと。

色: 文字色と同じ

- ・スタイルを変えたい文字列は、タグで囲う。
- ・スタイルを複数指定するときは””のなかで並べて書けば良い。
ここは赤文字、黒枠になる

課題

- ・ 次の HTML を書け。
ファイル名 : 1_6_textstyle.html

```
1 <html>↓
2 <head>↓
3 <title>文字のスタイル</title>↓
4 </head>↓
5 <body>↓
6 <div>↓
7 <span style="background-color:silver;">↓
8   ここではspanタグにスタイル属性を書いたが、 ↓
9   <span style="color:red">他のタグの中にも書くこともできる</span>。<br>↓
10  もしそうすると、そのタグの ↓
11  <span style="border:solid 1px black;">中身全て</span>↓
12  にその属性が適用される。 ↓
13 </span>↓
14 </div>↓
15 </body>↓
16 </html>↓
17 [EOF]
```

- ・ 次のように表示されるように書き足せ。

重要なポイント！

ここではspanタグにスタイル属性を書いたが、他のタグの中にも書くこともできる。
もしそうすると、そのタグの 中身全て にその属性が適用される。

ここで、「重要なポイント！」のところの文字の大きさは<h1>で指定されている。
背景色の水色はlightblueである。

1-7 領域のスタイル

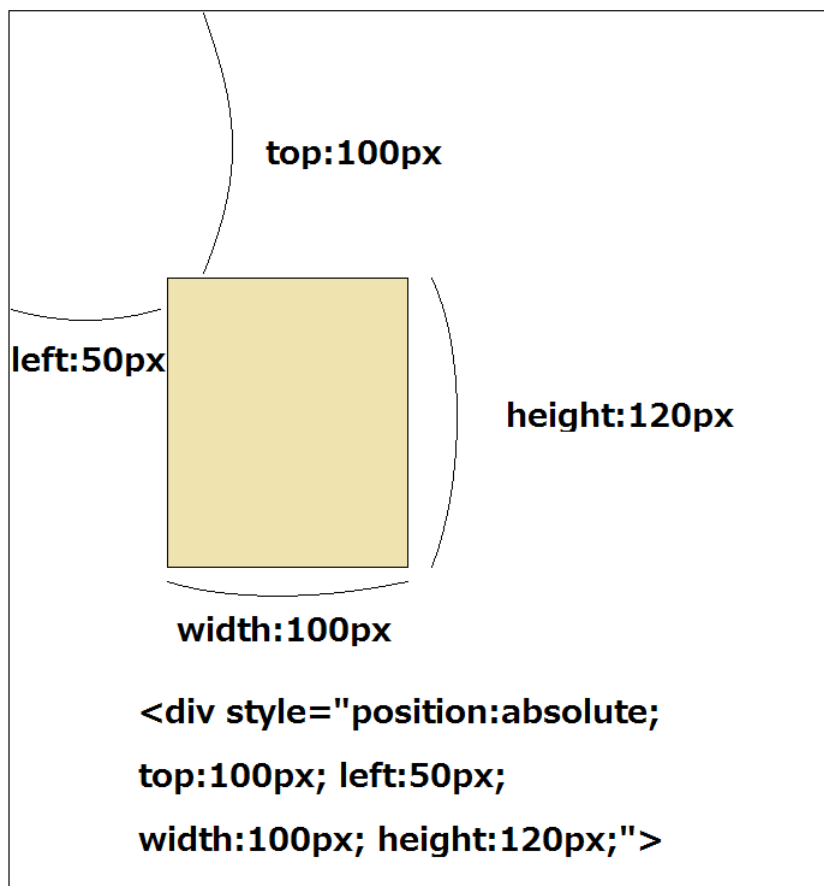
ポイント

- ・ 1-6 で学んだ文字のスタイルは、<div>タグのような領域を表すタグにも使える。
- ・ <body>, <p>, <div>などのタグでは、さらに次のようなスタイルが使える。

"background-image: url('画像ファイル');" … **背景画像**を指定（並べて表示される）

"width: 100px; height:200px;" … **横幅**（width）と**高さ**（height）の指定

"position:absolute; top:10px; left:30px;" … **上からの距離**（top）と**左からの距離**（left）



課題

- ・次のような小さい画像を作って、img1_7.png と名付けて保存しよう。



- ・次の HTML を書こう。

ファイル名：1_7_divstyle.html

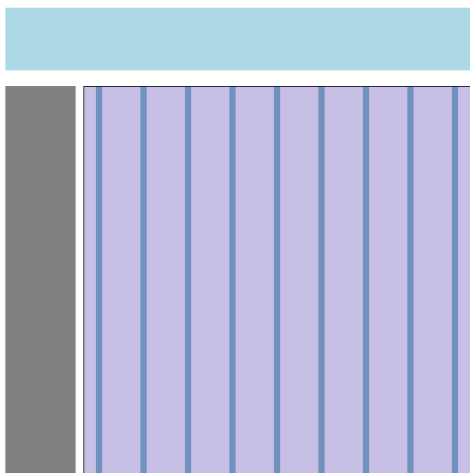
```
1 | <html>↓
2 |   <head>↓
3 |     <title>領域のスタイル</title>↓
4 |   </head>↓
5 |   <body>↓
6 |     <div style="width:500px; height:500px; position:absolute; top:100px;↓
7 |       left:100px; background-image:url('img1_7.png'); border:solid 1px black;">↓
8 |   </div>↓
9 | </body>↓
10| </html>↓
11|[EOF]
```

背景として、作った小さい画像が縦横にしき詰められたものが生成されたことが分かるだろう。
横幅、縦幅、上からの距離が正しく反映されているか確認しよう。

- ・次に、div の中に文章を書いてみよう。

```
<div ... >
  <p>ここに文章を書く</p>
</div>
```

- ・次のように表示されるように、左側と上側に新しい<div>領域を 2 つ作ろう。



これは、上にタイトル、左にメニュー、右下に本文を入れる、よくある web ページの形式である。
上手くいかないときは「要素の検証」を試してみるといい。

1-8 セレクタとスタイルシート

ポイント

- ・ 1-7 で分かったと思うけど、タグの中にスタイルを書くのだと長くなりすぎてしまう。
ここで、スタイルだけ別のところに分けて書く方法がある。
- ・ スタイルを指定したいタグには id 属性を付ける。
`<div id="id 名"> … </div>`
- ・ この id に対するスタイルは<head>タグの中で次のように書く。この方法を id セレクタという。

```
<style>
    #id 名 {
        color:blue;
        ...
    }
</style>
```

このようにスタイルだけまとめて書かれた部分をスタイルシートという。

課題

- ・ 1-7 の課題で作った 3 つの div について、それぞれ別の id 名をつけて、スタイルを<head>タグの中
に書け。例えば、背景のある div だけの時は次のように書ける。

ファイル名：1_8_selector.html

```
1 <html>↓
2 <head>↓
3 <title>セレクタとスタイルシート</title>↓
4 <style>↓
5 #id1 {↓
6     width:500px;↓
7     height:500px;↓
8     position:absolute;↓
9     top:100px;↓
10    left:100px;↓
11    background-image:url('img1_7.png');↓
12    border:solid 1px black;↓
13 }↓
14 </style>↓
15 </head>↓
16 <body>↓
17 <div id="id1">↓
18 </div>↓
19 </body>↓
20 </html>↓
21 [EOF]
```

ここがスタイルシート

この#id1{ … }の次の行に、同じように#id2{ … }と#id3{ … }を作れば良い。

1-9 練習問題 2

スタイルの技術を使って、整った web ページを作ってみましょう。(1_9_practice2.html)

次の HTML を土台にして下の画像のようなページにしよう。(青でなくても好きな色を使って良い)

margin:0px auto; は、上下の余白を 0 に、左右の余白を自動設定(中央寄せ)という意味。

また、特定方向の枠線を変更するには、

border:solid 1px black;

border-left:solid 5px blue; (左の枠線だけ 5px 青色にする)

のように、次に一行書く。左なら border-left, 右なら right, 上なら top, 下なら bottom である。

```
1 <html>↓
2 <head>↓
3 <title>練習問題 2</title>↓
4 <style>↓
5 </style>↓
6 </head>↓
7 <body>↓
8 <div style="position:relative; width:600px; margin:0px auto;">↓
9 <div id="header">↓
10 K会冬期講習2011↓
11 </div>↓
12 <div id="navi">↓
13 1 日目↓
14 </div>↓
15 <div id="honbun">↓
16 練習問題 2 ↓
17 </div>↓
18 </div>↓
19 </body>↓
20 </html>↓
21 [EOF]
```

おわったら一休みしましょう 且c(°▽*)ウマ-



1-10 リスト

ポイント

- ・メニューや目次など、項目を列挙するときにはとタグを使う。
- ・~がリストの始まりと終わりを表す。~はリストの一つの項目を表す。
- ・リストの中にリストが入っていても良い。

課題


- ・次のHTMLを書こう。

ファイル名：1_10_list.html

```
1 <html>↓
2 <head>↓
3 <title>リスト</title>↓
4 </head>↓
5 <body>↓
6 <div>↓
7 ↓
8 <ul>↓
9 <li>↓
10 A↓
11 </li>↓
12 <li>↓
13 B↓
14 </li>↓
15 <li>↓
16 C↓
17 </li>↓
18 </ul>↓
19 ↓
20 </div>↓
21 </body>↓
22 </html>↓
23 [EOF]
```

- ・次のように書き換えて、リストの中にリストを入れてみよう。

```
<li>↓
A↓
</li>↓
```



```
<li>↓
A↓
<ul>↓
<li>a1</li>↓
<li>a2</li>↓
</ul>↓
</li>↓
```

- ・次のように表示されるように改良しよう。
一日目の各項目は、自分が作ったページへのリンクにする。

- | |
|-----|
| 一日目 |
|-----|

 - [はじめの一步](#)
 - [HTMLとは](#)
 - [文の強調と区切り](#)
 - [リンクと画像](#)
- | |
|-----|
| 二日目 |
|-----|

 - javascriptの仕組み
 - セクタ

1-11 表

ポイント

- ・表を作るときは<table>, <tr>, <th>, <td>を使う。
- ・要素を縦横に整列させたいときにも使える。

<table> … 表を作る

<tr> … このタグで囲われた部分が表の中の一行になる

<th> … 見出しのマスはこれで囲う

<td> … 普通のマスはこれで囲う

課題

- ・次のHTMLを書こう。

ファイル名：1_11_table.html

```

1 <html>↓
2 <head>↓
3 <title>表</title>↓
4 </head>↓
5 <body>↓
6 <div>↓
7   ↓
8   <table>↓
9     <tr>↓
10       <th>タグ</th>↓
11       <th>意味</th>↓
12     </tr>↓
13     <tr>↓
14       <th>th</th>↓
15       <td>見出し</td>↓
16     </tr>↓
17   </table>↓
18   ↓
19 </div>↓
20 </body>↓
21 </html>↓
22 [EOF]
```

学年	クラス	名前
1	2	〇〇 〇〇

- ・次のように表示されるようにしよう。

タグ 意味
 th 見出し
 tr 1行
 td 1マス

・画像を並べる

表の各マスには画像を入れることもできる。画像を入れるときは、

`<td></td>`

のようにする。

つぎのように表示されるように書きかえてみよう（画像の入手法は授業中に指示する）。



・表に線を引く

スタイルのところにでてきた border を使う。

1.<head>タグの中に、1-8 でやったように<style>タグを追加する。

2.<style>タグの中に、次のように書く。

```
td {  
  border:solid 1px black;  
}  
th {  
  border:solid 1px black;  
}
```

この書き方は、

「<td>タグはすべて 1px の黒線で囲う」

「<th>タグはすべて 1px の黒線で囲う」

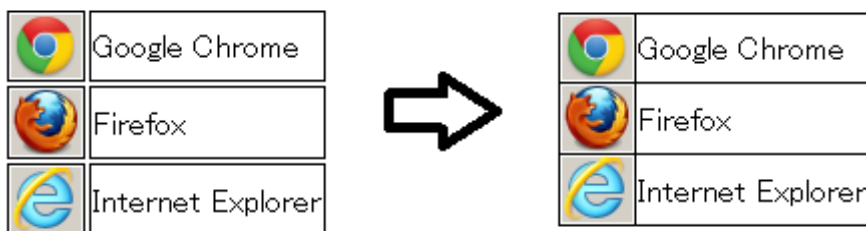
という意味である。

3.表の中の td と th のマスが黒線で囲まれるはずである。でもこのままだとなんか変だ。

そこで、もう一つ、次のスタイルを書き加えよう。

```
table {  
  border-collapse:collapse;  
}
```

これは、「<table>タグの中の枠線は、2つ並べないでくっつけて書く」という意味である。



1-12 練習問題 3

今日学んだことを使って、次のページにあるような web ページを作ってみよう。

- ・（その前に）スタイルの指定方法の確認

1. タグの中に書く

<タグ名 style="スタイル"> … </タグ名>

2. id で指定する（この id を持つタグのみのスタイルが変わる）

```
#id 名 {  
    スタイル
```

```
}
```

（ここまで<style>タグの内側）

…

<タグ名 id="id 名"> … </タグ名>

3. タグ名で指定する（このタグ名を持つタグすべてのスタイルが変わる）

```
タグ名 {  
    スタイル
```

```
}
```

（ここまで<style>タグの内側）

…

<タグ名> … </タグ名>

- ・上の「スタイル」の部分の書き方

color:blue;

border:solid 1px black;

のように、

スタイル名：スタイル；

の形式で並べて書く。タグの中に書くときは普通は；の後に改行しない。

- ・次のようなページを作ってみよう（ファイル名：1_12_practice3.html）
ただし、この通りにならなくてもいいし、表は面倒なので全て書き写さなくて良い。
作りたいものがある人はそれを作っても良い。
枠線を破線にするには border:dotted 1px blue;のように、solid のところを dotted にする。

K会冬期講習2011

1日目

- ・はじめの一步
- ・HTMLとは
- ・文の強調と区切り
- ・リンクと画像
- ・練習問題1
- ・文字のスタイル
- ・領域のスタイル
- ・セレクトとスタイルシート
- ・練習問題2
- ・リスト
- ・表
- ・練習問題3

今日学んだタグ

タグ名	意味	タグ名	意味
html	この中身がHTML	span	文字列
head	文書全体の情報	div	領域
title	タイトル	a	リンク
body	本文	img	画像
p	段落	style	スタイルをまとめて指定
h1~h6	見出し	ul	リストである
b	太字	li	リストの項目
i	斜体	table	表である
br	改行	tr	表の一行
hr	水平線	th	見出しのマス
		td	普通のマス

今日学んだスタイル

スタイル名	指定方法	意味
color	blue, red, ...	文字色
background-color	blue, red, ...	背景色
background-image	url(‘ファイル名’)	背景画像(並べて表示される)
border	solid 1px 色	枠線
border-top	solid 1px 色	枠線(上側)
border-right	solid 1px 色	枠線(右側)
border-bottom	solid 1px 色	枠線(下側)
border-left	solid 1px 色	枠線(左側)
border-collapse	collapse	表の枠線を一重にする
position	absolute, relative	absoluteにすると、位置を指定できるようになる
top	10px, 100px, ...	画面上端、あるいは親divの上端からの距離
left	10px, 100px, ...	画面左端、あるいは親divの左端からの距離
width	10px, 100px, ...	領域の横幅
height	10px, 100px, ...	領域の縦幅
margin	10px	ほかの要素との距離
padding	10px	内部の要素との距離

時間の余った人は、CSS リファレンス <http://www.htmq.com/style/index.shtml> で margin と padding の項目を見て、使ってみよう。

Part 1 はここまでです。お疲れさまでした\(^o^)/

2 日目 Javascript の基本

1. Javascript の仕組み
2. HTML 要素の取り出し
3. イベント
4. スタイルを操作する
5. 練習問題 1
6. 変数
7. 計算
8. フォーム
9. 条件分岐
10. 練習問題 2
11. シンプルなメニュー
12. 画像のサムネイル

2-1 Javascript の仕組み

ポイント

- ・ Javascript は、HTML を書き換えることで web ページに動きをつける
- ・ HTML の書き換えは、タグ単位で行う
- ・ 「イベント」は HTML を書き換えるタイミングになる
イベントには、「マウスが重なった」、「ページの読み込みが終わった」、などがある

課題

- ・ 具体例を見てみよう

<http://www.eizo.co.jp/>

動きのある部分について、要素の検証をしながら観察してみよう。

マウスに反応して HTML が書き変わっているのが分かるはずだ。

- ・ 次の HTML を書いてみよう。

ファイル名：2_1_shikumi.html

```
1 | <html>↓
2 |   <head>↓
3 |     <title>Javascriptの仕組み</title>↓
4 |     <script>↓
5 |       function dispTime(){↓
6 |         var d = new Date();↓
7 |         var time = document.getElementById("time");↓
8 |         time.innerHTML = d.getHours() + ":" + d.getMinutes();↓
9 |       }↓
10 |    </script>↓
11 |  </head>↓
12 |  <body onload="dispTime()">↓
13 |    <div>↓
14 |      <p>現在時刻は<span id="time"></span>です！</p>↓
15 |    </div>↓
16 |  </body>↓
17 | </html>↓
18 | [EOF]
```

- ・この HTML の解説(今はまだ理解できなくて良いが、ざっと目を通してほしい)

12 行目 onload="dispTime()"

onload は「ページが読み込まれたら」というイベントである。

onload="dispTime()"と書くと、ページが読み込まれたタイミングで

dispTime という名前の Javascript 命令が実行される。

最後の()は、dispTime が命令であることを表している。

4~10 行目 <script> ~ </script>

ここに Javascript が書かれていることを表す。

5~9 行目 function dispTime(){ ... }

dispTime という名前の**命令**を作っている。命令の内容は{ ... }の部分である。

6 行目 var d = new Date();

これは、次の 2 文に分解できる。

var d; ... d という名前の**変数**を作る

d = new Date(); ...変数 d に時刻の情報を表す**オブジェクト**を新しく生成して入れる

(この部分については今回の講習では深くは取り扱わない)

(用語の解説)

- ・変数

データを格納する**箱**のようなものである。

6 行目では、d という名前の箱を作って、new Date()から返ってきた

結果を箱にしまっている。

箱の中身は再利用したり、書き換えたりすることが出来る。

- ・命令

コンピュータにさせる仕事のこと。文字を書いたり絵を書いたりいろいろある。

変数と命令はカッコ () で区別できる。カッコがあるのが命令。

命令によっては**引数**をとったり、結果を返したりするものがある。

たとえば x+y を計算する命令は、引数が x と y で、足し算した結果を返す。

7 行目 `var time = document.getElementById("time");`
time という変数を作り、そこに命令 `getElementById("time")` の結果を入れる。
この命令は、web ページの本文を表すオブジェクト「**document**」が持っている命令で、**getElementById** という名前である。
("time") は、この命令に "time" という文字列を **引数** として渡すという意味で、
結果として **id が time の HTML タグ** を表すオブジェクト返ってくる。

14 行目 ``
id が time の HTML タグとはここである。

8 行目 `time.innerHTML = d.getHours() + ":" + d.getMinutes();`
time.innerHTML は、14 行目の `` タグの内側の部分を表す。最初は何も入っていない。
この部分が式の右辺の結果に書き換わる。
右辺は、3 つの文字列をくっつけたものである（この + は足し算でなく文字列の連結）。
時刻情報を表すオブジェクト d は命令 `getHours()` と `getMinutes()` を持っていて、
それぞれ時間、分を結果として返す。

まとめると、ページが読み込まれると（onload）、dispTime 命令が実行される。
dispTime 命令は、id が time となっている HTML タグの内部を、時間：分のデータで書き換える。
結果、「現在時刻は 00:00 です！」のように表示される。

（用語の解説）

・オブジェクト

変数と命令を集めたものと思えば良い。
オブジェクト A が変数 a, b, c と命令 f(), g() を持っていたとする。
このときオブジェクト A の変数 a は、**ドット** をつけて A.a と書く。
オブジェクト A の命令 f() を実行したいときは、A.f() と書く。

ところでオブジェクトが持っている変数がオブジェクトでもよい。
そういうときは A.B.C.f() のように多段階になることもある。
ところで、HTML のタグはオブジェクトとして扱われる。

`<html>.<head>`

`<html>.<body>.<div>.<p>.`

のような階層構造がある（実際はちょっと違う）。

2-2 HTML 要素の取り出し

ポイント

- ・ HTML 中の書き換えたい部分は id 属性で指定することが出来る。
 <タグ名 id="id 名"> (書き換えたい部分) </タグ名>
- ・ ある id 名を持つタグを javascript 上で**オブジェクト**として取り出すことで、書き換えを行う。
 var obj = document.getElementById("id 名");
 このとき obj には<タグ名 id="id 名"> … </タグ名>の範囲の情報が入っている。
- ・ 取り出したタグに挟まれた部分の HTML は
 obj.innerHTML
 で表される。これを書き換えれば、HTML が書き換わるというわけだ。
- ・ 書き換えはイコール=記号をつかって行う。
 obj.innerHTML = 書き換えたい文字列;

課題

- ・ 2-1 の HTML を改造して、次のように表示されるようにしよう。
 ファイル名：2_2_selector.html

今日は？年？月？日です。
現在時刻は？時？分です！

(？の部分には今の日付が入る)

ここで、年を取得するには getFullYear()命令、月は getMonth()命令、日は getDate()命令である。
getMonth()命令は実際の月から 1 を引いた数字を返すので、

 d.getMonth()+1
としてあげよう。

- ・ innerHTML には HTML タグが入っていても良い。？時？分の？を太字にしてみよう。
 ヒント：time.innerHTML = "" + …

2-3 イベント

ポイント

- ・ HTML を書き換えるタイミングは、「イベント」で指定することが出来る。
- ・ イベントには次のようなものがある。
 - onload … ページが読み込まれた
 - onclick … クリックされた
 - onmouseover … マウスが乗った
 - onmouseout … マウスが離れた
- ・ onclick, onmouseover, onmouseout は div などのタグに対してつけることができる。
`<div onmouseover="マウスが乗ったときに実行したい命令()">`
- ・ img タグの src 属性 (画像ファイル名) は、
`img = getElementById("image");`
`img.src = "画像ファイル名";`
のように、.src を使って操作する。

課題

- ・ onmouseover と onmouseout を使った次の HTML を書いてみよう。
ファイル名: 2_3_event.html

```
1 <html>↓
2 <head>↓
3 <title>イベント</title>↓
4 <script>↓
5     function changeAB(){↓
6         var img = document.getElementById("image");↓
7         img.src = "img2_3_B.png";↓
8     }↓
9 </script>↓
10 </head>↓
11 <body>↓
12 <div>↓
13 ↓
14 </div>↓
15 </body>↓
16 </html>↓
17 [EOF]
```

マウスを乗っけると画像が切り替わるはずだ。

- ・ マウスが離れたら画像が元に戻るようにしよう。
img タグのなかに onmouseout="changeBA()" を付け加えて、changeBA 命令を changeAB を真似して書けば良い。

2-4 スタイルを操作する

ポイント

- ・ マウスをかざすと色が変わる、とか、クリックすると消える、といった動作は HTML タグのスタイルを Javascript から書き換えることで実現する。
- ・ スタイルの書き換えは、「HTML 要素.style.スタイル名 = スタイルの内容;」の形式で行う。つまり、
 <div id="target"> (書き換えたい内容) </div>
 に対してスタイルを変えたければ、
 var t = document.getElementById("target");
 として HTML 要素を t に格納した後、
 t.style.color = "red";
 などとすると、スタイルを変更できる。ちなみにこれは文字色を赤にしている。
- ・ スタイル名には次のようなものがある。
 color, border, top, left, width, height … 昨日やったのと同じ意味 (1-12 参照)
 backgroundColor … 昨日やった background-color と同じ意味 (変数名には-が使えない)
 display … "none"を入れると消える。"block"を入れると出現する。

課題

- ・ 次の HTML を書け。(ファイル名: 2_4_style.html)

```
1 | <html>↓
2 |   <head>↓
3 |     <title>スタイルを操作する</title>↓
4 |     <script>↓
5 |       function click1(){↓
6 |         var d = document.getElementById("div1");↓
7 |         d.style.color = "red";↓
8 |       }↓
9 |       function click2(){↓
10 |        var d = document.getElementById("div2");↓
11 |        d.style.display = "none";↓
12 |      }↓
13 |    </script>↓
14 |  </head>↓
15 |  <body>↓
16 |    <div>↓
17 |      <div id="div1" onclick="click1()">div1</div>↓
18 |      <div id="div2" onclick="click2()">div2</div>↓
19 |    </div>↓
20 |  </body>↓
21 | </html>↓
22 | [EOF]
```

div1 をクリックしたら赤色になり、div2 をクリックしたら消えるはずである。

2-5 練習問題 1

- ・ 2-4 の HTML を改造しよう。（ファイル名：2_5_practice1.html）

まずは、div1 をクリックしたら、上から 500、左から 500 の位置にジャンプするようにしよう。
このとき、

```
d.style.position = "absolute";
```

の一文を追加する必要があることに注意（これを書かないと位置を動かせない）。

次に、div2 の上にマウスが乗っただけで、div2 が消えるようにしよう。

- ・ 次のような表を作れ。（ファイル名：2_5_practice1_2.html）



2×2 の表

マウスが離れていると、どちらの行も白

マウスが重なった行（<tr>～</tr>）が水色（"lightblue"）になる。

マウスが離れると元に戻る。

1 日目に表まで終わってない人は、上下に分かれた 2 つの div 領域でもよい。

ヒント：

「上の行にマウスが乗った時」、「上の行からマウスが出た時」、「下の行にマウスが乗った時」、「下の行からマウスが出た時」の 4 つの命令が必要である。

ここでひと休みしよう(*^_^*)

2-6 変数

ポイント

- ・ **変数**はデータを入れる箱である。
- ・ データには「数」、「文字」などがある。文字は” ”で囲ってあらわす。
100 と”100”は違う。100 は数としての百で、”100”はイチゼロゼロという文字列である。
- ・ 変数を作るには var を使う。

```
var a;
```

- ・ 変数にデータを入れるときは、=を使う。

```
a = 100;
```

- ・ すでに何かが入っている変数にデータを入れると、もともと入っていたデータは消える。
- ・ 中括弧 { } の内側で作った変数は、その中だけで使える。外側では使えない。

```
var a;
```

```
function func1(){
```

```
    var b;
```

```
    ...
```

```
}
```

```
function func2(){
```

```
    ここでは変数 a は使えるが、func1 の b は使えない。
```

```
}
```

課題

- ・ 次の HTML を書こう。（ファイル名：2_6_var.html）

```
1 <html>↓
2 <head>↓
3 <title>変数</title>↓
4 <script>↓
5     var n = 0;↓
6     function add(){↓
7         var d = document.getElementById("num");↓
8         n = n+1;↓
9         d.innerHTML = n;↓
10    }↓
11 </script>↓
12 </head>↓
13 <body>↓
14 <div>↓
15     <div onclick="add()">クリック！</div>↓
16     <div id="num">0</div>↓
17 </div>↓
18 </body>↓
19 </html>↓
20 [EOF]
```

クリック！をクリックすると数が1 ずつ増えるはずである。

n = n+1; という文は、n に n+1 を入れるという意味だから、変数 n は 1 増える。

それを二番目の div の innerHTML に書き込んでいるから、クリックするたびに数が増えるのである。

2-7 計算

ポイント

- ・変数は、データの**記憶**のためと、データを用いた**計算**のために使われる。
- ・「数」に対しては次のような計算ができる。
 - 足し算： +
 - 引き算： -
 - かけ算： *
 - わり算： /
 - 割った余り： % (5%2 は 1, 10%3 は 1, 50%23 は 4)
 - カッコ： () (算数のカッコと同じ意味。1+2*3 は 7 だが(1+2)*3 は 9 である)
- ・「文字」に対しては次のような計算ができる。
 - 連結： + ("abc" + "def" は "abcdef" になる)
- ・「数」と「文字」は**変換可能**である。String と parseInt という命令を使う。
 - String(100) これは文字の"100"になる
 - parseInt("100") これは数の 100 になる
 - parseInt("100px") これも数の 100 になる。実はこれが大事。
 - parseInt("a123") これはエラーになるこの例のように、parseInt は、カッコの中が数字+文字の場合は文字を無視してその数字を返す。
- ・web ページで数を扱う際に最もよく出てくるのが、style の top, left, width, height である。そこでは obj.style.top = "100px" といった感じで、**文字列**で記録されている。従って、10px 下に移動しようとしたら、
 - obj.style.top = parseInt(obj.style.top) + 10;としなければいけない。ここで、右辺を"110px"にしなくていいか不思議かもしれないが、これはコンピュータが勝手にやってくれる！

課題

- ・ 2-6 のプログラムを改造して、クリック！をクリックすると、数字が
1, 2, 4, 8, …
と二倍二倍になっていくようにしよう。（8 行目を書きかえる）
- ・ さらに改造して、クリックすると、数字が
1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, …
と 4 まで進んだら 0 に戻るようにしよう。これは割った余り「%」記号を使うとできる。
- ・ 今度は文字にして、
a, aa, aaa, aaaa, aaaaa, …
と a が一個ずつ増えていくようにしてみよう。（5, 8, 16 行目を書きかえる）
- ・ 今度はクリックすると、数字が
1, 1, 2, 3, 5, 8, 13, 21, 34, … （フィボナッチ数列。前の 2 つを足すと次の数になる）
の順で増えていくようにしよう。これは変数をもう一つ使う必要がある。

2-8 フォーム

ポイント

- ・キーボードから文字や数を入力してもらうには、「フォーム」を使う。
- ・フォームはタグを使って作る。

```
<input id="input1">
```

- ・フォームの内容は value 属性で参照できる。

```
var i = document.getElementById("input1");
```

としたとき、**i.value** で input1 のフォームに入力されている情報

100
200
0

フォームの例

がわかる。

- ・ついでに、input タグの type 属性を button にすることで、**ボタン**を作ることにも出来る。
value 属性がボタンに書かれる文字で、onclick イベントで押された時の動作を決める。

```
<input type="button" value="押す" onclick="func()">
```

押す

ボタンの例

課題

- ・次の HTML を書け。（ファイル名：2_9_form.html）

```

1 <html>↓
2   <head>↓
3     <title>フォーム</title>↓
4     <script>↓
5       function change(){↓
6         var r = document.getElementById("red");↓
7         var g = document.getElementById("green");↓
8         var b = document.getElementById("blue");↓
9         var d = document.getElementById("d");↓
10        d.style.backgroundColor="rgb("+r.value+", "+g.value+", "+b.value+")";↓
11      }↓
12    </script>↓
13  </head>↓
14  <body>↓
15    赤<input id="red"> <br>↓
16    緑<input id="green"> <br>↓
17    青<input id="blue"> <br>↓
18    <input type="button" value="押す" onclick="change()"> <br>↓
19    <div id="d" style="width:200px; height:200px"></div>↓
20  </body>↓
21 </html>↓
22 [EOF]
```

うまくいけば、フォームが3つとボタンが1つ表示される。

フォームには0~255の数字をそれぞれ書く。この数字の意味は、フォームの左に書いてある色（光の三原色）の強さを表す。書いた後にボタンを押すと、下に3つを混ぜた色が表示される。

例えば、赤 255, 緑 0, 青 0 ならきれいな赤。100, 0, 100 なら紫になる。

・この HTML の解説

10 行目の文は 19 行目にある div の背景色を変更している。

背景色は、今までは blue, red などの英語で指定していたが、ここでは

`d.style.backgroundColor="rgb(100, 200, 0)";`

のような、赤、緑、青の混ぜ具合で指定している。1 つ目の数字が赤、2 つ目が緑、3 つ目が青。

3 つの色をフォームから取り込むのだが、各フォームの情報は 6~8 行目の命令によって変数 r, g, b に入っており、フォームの中に書かれている文字列は r.value のように.value を付けることで取り出すことができる。

そこで、7 つの文字列

`rgb(r.value , g.value , b.value)`

を+を使って連結することで、`"rgb(100, 200, 0)"`のような文字列を作り出している。

・うまく 3 つの数字を組み合わせて次の色を作ってみよ。

白

黄色

灰色

水色

茶色

・次のような HTML を書け。

2 つのフォームと 1 つのボタンがあり、ボタンを押すと 1 つ目のフォームの中身を高さ、2 つ目のフォームの中身を横幅とした長方形をボタンの下に表示する。

縦

横

2-9 条件分岐

ポイント

・入力されて文字が〇〇だったら…、数がx以上だったら…、偶数回クリックされたら…、みたいな場合分けのことを**条件分岐**という。

- ・条件分岐には if 構文を使う。

```
if(条件 A){  
    Aがみたされているときに実行すること  
}  
else if(条件 B){  
    Aがみたされていなくて、Bがみたされているときに実行すること  
}  
else{  
    AもBもみたされていないときに実行すること  
}
```

- ・「条件」は、次の記号を使って書く。

(変数 a, b が数のとき)

a == b	a と b が等しい
a != b	等しくない
a > b	a が b より大きい
a >= b	a が b 以上
a < b	小さい
a <= b	以下

(変数 a, b が文字のとき)

a == b	a と b が同じ
a != b	違う

(A, B が条件の時)

A && B	A と B が両方みたされている
A B	A, B のどちらかがみたされている

- ・条件の例

a == "abc"	文字 a が "abc" だったら
5 <= a && a <= 10	a が 5 以上 10 以下だったら (5 <= a <= 10 とは書けないので注意)
a+b == "hello"	文字列 a, b を連結すると "hello" になるなら

課題

- ・2つの違う画像を用意しよう。一目目の google と yahoo の画像で良い。
- ・次の HTML を書こう。ただし、9, 12, 21 行目の画像ファイル名は、今用意した画像名にする。
ファイル名：2_9_if.html

```

1 <html>↓
2 <head>↓
3 <title>条件分岐</title>↓
4 <script>↓
5     var a = 0;↓
6     function change(){↓
7         var image = document.getElementById("img");↓
8         if(a==0){↓
9             image.src = "img2_9_2.png";↓
10            a = 1;↓
11        }else{↓
12            image.src = "img2_9_1.png";↓
13            a = 0;↓
14        }↓
15    }↓
16 </script>↓
17 </head>↓
18 <body>↓
19 <div>↓
20     <input type="button" value="切り替え" onclick="change()"> <br>↓
21      <br>↓
22 </div>↓
23 </body>↓
24 </html>↓
25 [EOF]

```

20行目でボタンを作っている。このボタンがクリックされると(onclick イベント)、change() 命令が呼ばれる。change は6~15行目で作られていて、変数 a の値によって場合分けされている。変数 a は5行目で作られていて、最初は0が入っている。

変数 a が0のときは1枚目の画像、1の時は2枚目の画像、となっており、たとえば a==0 のときに change() 命令が実行されると、画像を2枚目のに切り替えて(9行目)、aを1にしている(10行目)。

- ・この HTML を改造して、3枚の画像が切り替わるようにしよう。

2-10 練習問題 2

・BMI 指数とは、「体重(kg)÷(身長(m)×身長(m))」で計算される数値で、18.5 未満でやせ気味、25 以上でふとり気味、18.5~25 が標準的という目安である。

ここで、次のようなページをつくろう。(ファイル名: 2_10_practice2.html)

身長(m):

体重(kg):

BMI は 17.35891932836763 です。
やせ気味？

フォームが 2 つ (身長と体重)、ボタンが 1 つあり、ボタンを押すと BMI 値と、やせ気味、普通、ふとり気味の目安を表示する。

フォーム `f` の中身は `f.value` で使うことができるが、このままだと文字なので、

`parseFloat(f.value)`

と書くことで、数に変換する。

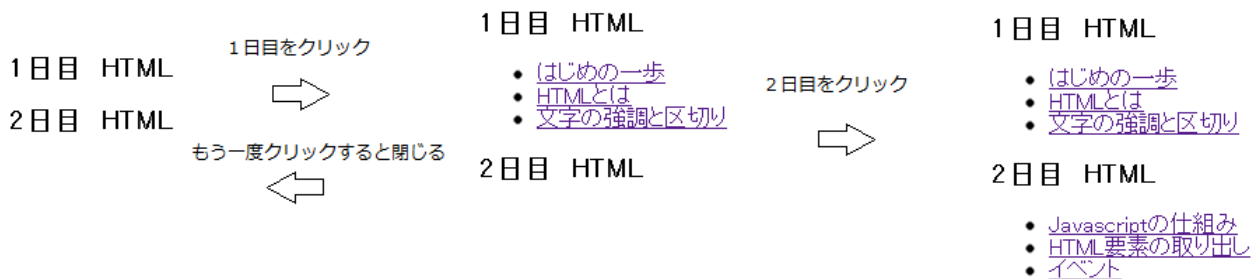
がんばれべ(。・ω・)ノ

2-11 シンプルなメニュー

2-11、2-12 では、ここまで勉強してきたことを使って実用的なプログラムを作る。

まずは次のようなシンプルなメニューを作ってみよう。

このメニューは項目をクリックすると、その中身が開いたり閉じたりする。



上の例では中身は3つしか作ってないが、これが10個とかになると、最初から中身を表示すると長くなりすぎてしまうので、このようなスクリプトを使ってコンパクトに収まるようにするのが普通である。

上の例は、中身にリスト（1-9 参照）を使っているが、1 日目にリストまで進んでない人は div を使ってやっても良い。

あるあるいは<div>領域について、スタイルが display:none となっているとその領域は表示されない。display:block だと表示される。

この領域の id を test とすると、この領域（最初は display:none 表示されていない）を表示するスクリプトは、

```
var t = document.getElementById("test");  
t.style.display = "block";
```

となる。

自信のある人は次のページのヒントを見ないでプログラムを書いてみよう。

- ・ できたら、クリックではなくマウスが乗ったらメニューが開いて、離れたら閉じるように改良しよう。

- ・ シンプルなメニューのプログラム例
空欄を埋めよう。

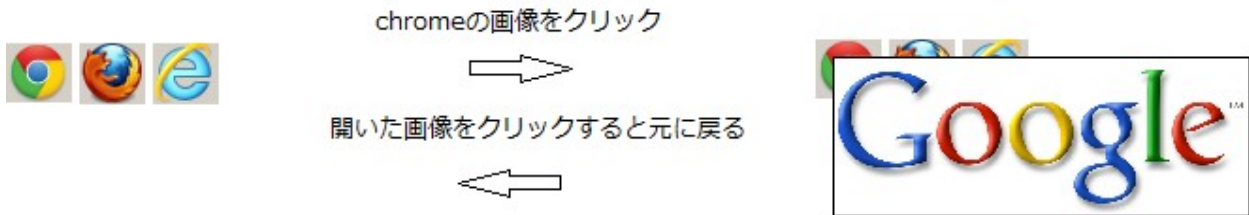
```

1 <html>↓
2 <head>↓
3   <title>シンプルなメニュー</title>↓
4   <script>↓
5     function change1(){↓
6       var p = document.getElementById(" ");↓
7       if(p.style.display==""){↓
8         p.style.display="";↓
9       }else{↓
10        p.style.display="";↓
11      }↓
12    }↓
13    function change2(){↓
14      
15    }↓
16  }↓
17 </script>↓
18 </head>↓
19 <body>↓
20   <div>↓
21     <h3 onclick="change1()">1日目 HTML</h3>↓
22     <ul id="part1" style="display:none">↓
23       <li><a href="1_1_intro.html">はじめての一步</a></li>↓
24       <li><a href="1_2_html.html">HTMLとは</a></li>↓
25       <li><a href="1_3_kyouchou.html">文字の強調と区切り</a></li>↓
26     </ul>↓
27     ↓
28     <h3 onclick="change2()">2日目 Javascript</h3>↓
29     <ul id="part2" style="display:none">↓
30       <li><a href="2_1_shikumi.html">Javascriptの仕組み</a></li>↓
31       <li><a href="2_2_selector.html">HTML要素の取り出し</a></li>↓
32       <li><a href="2_3_event.html">イベント</a></li>↓
33     </ul>↓
34   </div>↓
35 </body>↓
36 </html>↓
37 [EOF]

```

このアドレスは自分が作ったファイル名にあわせて

2-12 画像のサムネイル



写真集などをつくる時、最初から写真を全部表示すると読み込みに非常に時間がかかってしまう。そこで、写真を小さくした画像を作っておいて、最初の画面ではそれを並べたものを見せ、見る人が見たい写真をクリックしたらその写真の大きいバージョンを表示するようにすると、使いやすい写真集を作ることが出来る。このような方法を「サムネイル」という。

上の例のように小さい画像を並べて表示し、それぞれクリックすると大きいものが開き、開いた画像をクリックするとそれが閉じるようなプログラムを作ろう。

2-11 のメニューと似たような感じで作れるので、ノーヒントでやってみてほしい。画像は自分で適当に用意せよ。

3 日目 繰り返しの処理

1. タイマー
2. タイマーを使ったアニメーション
3. ループ
4. 二重ループ
5. 練習問題 1
6. class
7. `getElementsByClassName`
8. 配列
9. 練習問題 2
10. 乱数
11. 応用

3-1 タイマー

ポイント

- ・ 指定した時間が経った後に何かをするときは**タイマー**を使う。
`setTimeout("命令の名前()", 何ミリ秒後);`

課題

- ・ 次の HTML を書こう。

```

1 <html>↓
2 <head>↓
3 <title>タイマー</title>↓
4 <script>↓
5     function setTimer(){↓
6         setTimeout("dispImg()", 5000);↓
7     }↓
8     function dispImg(){↓
9         var img = document.getElementById("div1");↓
10        img.style.display = "block";↓
11    }↓
12 </script>↓
13 </head>↓
14 <body onload="setTimer()">↓
15     <div id="div1" style="display:none;">↓
16         ↓
17     </div>↓
18 </body>↓
19 </html>↓
20 [EOF]
```

好きな画像ファイルを使おう

14 行目で `onload="setTimer()"` としているので、ページが開かれたらすぐに `setTimer()` 命令(5~7 行目)が実行される。この命令では、6 行目にあるように、5000 ミリ秒 = 5 秒後に `dispImg()` 命令(8~11 行目)を実行するように書かれている。よって、ページを開いてから 5 秒立ったときに画像が表示される。

- ・次の HTML を書こう。

```

1 <html>↓
2   <head>↓
3     <title>タイマー2</title>↓
4     <script>↓
5       var a=0;↓
6       function timer(){↓
7         a = a+1;↓
8         var t = document.getElementById("t");↓
9         t.innerHTML = a;↓
10        setTimeout("timer()", 100);↓
11      }↓
12    </script>↓
13  </head>↓
14  <body>↓
15    <div>↓
16      <input type="button" value="start" onclick="timer()"> <br>↓
17      <span id="t">0</span>↓
18    </div>↓
19  </body>↓
20 </html>↓
21 [EOF]

```

今度はボタンを押したら timer() 命令が呼ばれる。10 行目で timer 命令自身を 0.1 秒後に予約しているところに注目しよう。こうすることで、0.1 秒おきに timer 命令が実行され、その都度 id が t の領域に表示される数字が 1 ずつ増えていくことになる。

- ・このプログラムにストップボタンもつけてみよう。

3-2 タイマーを使ったアニメーション

ポイント

- ・一定時間ごとに位置をちょっとずつ動かしていくと、アニメーションが作れる。

課題

- ・次の HTML を書こう。17 行目の画像ファイル名は、自分の好きな画像にせよ。

```
1 <html>↓
2 <head>↓
3 <title>アニメーション 1</title>↓
4 <script>↓
5   var a=0;↓
6   function timer(){↓
7     a = a+1;↓
8     var image = document.getElementById("img");↓
9     image.style.top = a;↓
10    setTimeout("timer()", 10);↓
11  }↓
12 </script>↓
13 </head>↓
14 <body>↓
15 <div>↓
16   <input type="button" value="start" onclick="timer()"> <br>↓
17   <div id="img" style="position:absolute"></div>↓
18 </div>↓
19 </body>↓
20 </html>↓
21 [EOF]
```

9 行目に注目すると、timer 命令は 17 行目にある div の上からの距離を変数 a にしていることがわかる。

10 行目を見ると、timer 命令は timer 命令自身を 0.01 秒後に予約するので、timer 命令は 1 秒間に 100 回実行されることが分かる。

さらに 7 行目を見ると、timer 命令が一回実行されるたびに a の値は 1 ずつ増えていく。

以上より、このプログラムは 17 行目の div を毎秒 100px のペースで下に向かって動かすことがわかる。

・このプログラムを改造して、最初透明だった画像が徐々に不透明になっていくようなプログラムをつくろう。

透明度は、style の一つの opacity で調整でき、今までの position や left と同じように image.style.opacity で変えることが出来る。

opacity は 0 のときに完全に透明で、opacity が 1.0 のときに完全に不透明になる。

(・次のプログラムは、マウスに画像がついてくるようなプログラムである。)

```

1 <html>↓
2 <head>↓
3 <title>アニメーション3</title>↓
4 <script>↓
5   var tx=0;↓
6   var ty=0;↓
7   var cx=0;↓
8   var cy=0;↓
9   function mv(){↓
10    tx = event.x;↓
11    ty = event.y;↓
12  }↓
13  function timer(){↓
14    var img = document.getElementById("image");↓
15    cy = ty;↓
16    cx = tx;↓
17    img.style.top = cy;↓
18    img.style.left = cx;↓
19    setTimeout("timer()", 50);↓
20  }↓
21 </script>↓
22 </head>↓
23 <body onload="timer()">↓
24   <div style="width:500; height:500; position:absolute;" id="div1" onmousemove="mv()">↓
25     ↓
26   </div>↓
27 </body>↓
28 </html>↓
29 [EOF]

```

24 行目の onmousemove は、「マウスが動いたら」というイベントである。マウスが動くたびに、9-12 行目に書かれた mv() 命令が実行される。

マウスに画像をついてこさせるのなら mv() 命令の中で画像の top と left を動かしてもよいのだが、そうすると画像の書きかえがたくさん起こってしまい、処理が遅くなる。ここで、タイマーを使う。

今変数 tx, ty は画像を動かす目標地点を、cx, cy は次に画像を動かすべき場所を記録する変数とする。mv 命令の中では、目標地点をマウスの座標(event.x, event.y にはマウスの x, y 座標が自動的に入っている)にセットする。一定時間ごとに呼ばれる timer 命令の中で、img の位置を動かしている。

・このままではマウスの場所と画像の中心とが少しずれているはずである。これを合わせよう。

・15, 16 行目をいじって、画像がゆっくりとマウスに近づいていくようにしてみよう。

ヒント：上のプログラムのままでは、画像は一瞬でマウスの座標に移動する。だから、15, 16 行目で cx, cy を動かすすぎないようにすれば、画像はゆっくりと近づいていくようになるだろう。

3-3 ループ

ポイント

- ・似たようなことを何回もしたいときは「for ループ」を使う。
- ・タイマーは一定時間ごとに実行していたが、ループは一気に実行される。
- ・for 文は次のように書く。

```
for(var i=0; i<n; i++){
    ...
}
```

このとき、変数*i*が作られ、*i*が 0, 1, 2, …, *n*-1 と変化しながら… が *n* 回実行される。

例：for(var i=0; i<5; i++)

i は 0, 1, 2, 3, 4 と変化する。

例：for(var i=10; i>0; i--)

i は 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 と変化する。

例：for(var i=1; i<10; i=i+3)

i は 1, 4, 7 と変化する。

例：for(var i=1; i<40; i=i*2)

i は 1, 2, 4, 8, 16, 32 と変化する。

課題

- ・for 文は同じようなものをたくさん作るのに適している。ここでは次の HTML を書いて、100 個の div を作ってみよう。

```
1 <html>↓
2 <head>↓
3 <title>ループ 1</title>↓
4 <script>↓
5     function func(){↓
6         var d1 = document.getElementById("div1");↓
7         var s = "";↓
8         for(var i=0; i<100; i++){↓
9             s = s + "<div>" + i + "</div>";↓
10        }↓
11        d1.innerHTML = s;↓
12    }↓
13 </script>↓
14 </head>↓
15 <body onload="func()">↓
16     <div id="div1">↓
17     </div>↓
18 </body>↓
19 </html>↓
20 [EOF]
```

- ・このプログラムの解説

func()命令は onload に指定されているので、ページが開かれたときに一回だけ実行される。

8~10 行目が for 文であり、i は 0, 1, 2, …, 99 と変化するはずである。

7 行目で作られている変数 s は文字列であり、最初は何も入っていない。

9 行目を見ると、for ループが回るたびに、s に「<div>i</div>」（i はそのときの変数 i の値）という文字列が追加されていくことが分かる。

これが 11 行目で dl.innerHTML に入れられるので、結局それぞれ 0~99 までの数字が書かれた 100 個の div 領域が作られることになる。

- ・このプログラムに入力フォームとボタンを付け（2-8 などを参照）、フォームに数を入力して、ボタンを押すと、入力した数だけ div 領域が生成されるようにせよ。

- ・このプログラムでは s という文字列に、ループが回るたびに”<div>5</div>”のような新たな文字列が追加されていったが、s を文字列ではなくて数の変数にして、毎回 i を足し算するようにすれば、0 から(フォームに入力した数-1)までの数の合計を求めることが出来るだろう。これをやるプログラムを書こう。これを使って $1 + 2 + \dots + 1000$ を求めよ。

3-4 二重ループ

ポイント

- ・ for 文のなかに for 文を入れると、「二重ループ」を作ることが出来る。これは二次元の表を作ったりするときに役に立つ。

課題

- ・ 次の HTML を書け。

```

1  <html>↓
2  <head>↓
3  <title>二重ループ 1</title>↓
4  <script>↓
5      function func(){↓
6          var kuku = document.getElementById("kuku");↓
7          var str = "";↓
8          for(var i=0; i<10; i++){↓
9              for(var j=0; j<10; j++){↓
10                 str += (i*j) + " ";↓
11             }↓
12             str += "<br>";↓
13         }↓
14         kuku.innerHTML = str;↓
15     }↓
16 </script>↓
17 </head>↓
18 <body onload="func()">↓
19     <div id="kuku"></div>↓
20 </body>↓
21 </html>↓
22 [EOF]

```

8-15 行目の for 文の中に、9-11 行目のもう一つの for 文が入っている。カウンターに使っている変数は、i と j で、もちろん 2 つの for 文で使い分けないとイケない。

このとき、外側のループが一回回る間に、内側のループは 10 回回っている。

よく最終的に出力された HTML と、プログラムとを見比べて、どの文がどれを書いているのかを考えてほしい。

ちなみに、10 行目にでてくる

```
str += (i*j) + " ";
```

というのは、

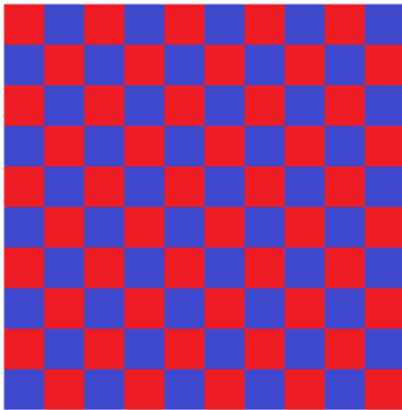
```
str = str + (i*j) + " ";
```

を略して書いたものである。

・1-11の「表」をやった人は、今のプログラムを<table>タグを使って書きかえてみよう。そうすると行と列がちゃんと揃うようになる。

0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8
0	2	4	6	8	10	12	14	16
0	3	6	9	12	15	18	21	24
0	4	8	12	16	20	24	28	32
0	5	10	15	20	25	30	35	40
0	6	12	18	24	30	36	42	48
0	7	14	21	28	35	42	49	56
0	8	16	24	32	40	48	56	64
0	9	18	27	36	45	54	63	72

・このプログラムを参考にして、次の模様を書いてみよう。（<table>を使わずともできる）



3-5 練習問題 1

- ・ 次のような x の y 乗を計算するプログラムを書け。

x の y 乗を計算します。
x:
y:

5 の 4 乗は 625 です。

- ・ 次のような、 x 以下の素数を全て計算するプログラムを書け。

x 以下の素数を全て計算します。
x:

2
3
5
7
11
13
17
19
23
29
31
37

ヒント：整数 p が素数であるとは、1 と p 以外に p を割り切る数がないということ。

「 a が p を割り切る」という条件は「 $p \% a == 0$ 」と書ける。

(・相性診断)

```

1 <html>↓
2 <head>↓
3 <title>練習問題1_3</title>↓
4 <script>↓
5   function calc() {↓
6     var n1 = document.getElementById("name1");↓
7     var n2 = document.getElementById("name2");↓
8     var d = document.getElementById("div1");↓
9     var a1 = 0;↓
10    var a2 = 0;↓
11    for(var i=0; i<n1.value.length; i++){↓
12      a1 = a1 + n1.value.charCodeAt(i);↓
13    }↓
14    for(var i=0; i<n2.value.length; i++){↓
15      a2 = a2 + n2.value.charCodeAt(i);↓
16    }↓
17    d.innerHTML = n1.value + "と" + n2.value + "の相性は" + (100-Math.abs((a1%100)-(a2%100))) + "です。";↓
18  }↓
19 </script>↓
20 </head>↓
21 <body>↓
22   相性占い。<br>↓
23   あなたの名前:<input id="name1"> <br>↓
24   相手の名前:<input id="name2"> <br>↓
25   <input type="button" value="診断" onclick="calc()"> <br>↓
26   <div id="div1"></div>↓
27 </body>↓
28 </html>↓
29 [EOF]

```

これは簡単な相性占いのプログラムである。

自分の名前と相手の名前を入力すると、相性値が 0~100 で診断される。

このプログラムは、名前を数字に変換して、それを比較することで相性値を計算する。

まず、変数 n1.value と n2.value に名前の文字情報が入っている。

n1.value.charCodeAt(i) という命令は、文字列 n1 の i 番目の文字の文字コード（すべての文字に割り当てられている数字のこと）を得る命令で、これを変数 a1 に足すという動作を n1 の長さ（n1.value.length）回だけくり返すことで、a1 には n1 に含まれる各文字の文字コードの和が格納される。a2 も同様である。

あとはこの a1, a2 の 2 つを適当に比較してあげれば相性値を計算できる。

上のプログラムでは、それぞれを 100 で割った余りを引き算し、その差の絶対値（Math.abs 命令）を 100 から引いたものを相性値としている。こうすると、自分同士の相性値は 100、自分と相手を入れ替えても同じ、大きい数が出やすい、といった利点がある（まあこんなことにこだわらなくてもいいのだが・・・）。

3-6 class

ポイント

- ・ここから3つの章では新しい概念が3つ登場する。混乱しないように慎重に読み進めよう。
- ・class を用いた HTML 要素の指定
- ・getElementsByName
- ・配列

課題

- ・次の HTML を書こう。

```

1 <html>↓
2   <head>↓
3     <title>class</title>↓
4     <style>↓
5       .blue{↓
6         background-color:blue;↓
7       }↓
8       .red{↓
9         background-color:red;↓
10      }↓
11    </style>↓
12  </head>↓
13  <body>↓
14    <div class="blue">これ</div>↓
15    <div class="blue">ら</div>↓
16    <div class="blue">は</div>↓
17    <div class="blue">blueクラス</div>↓
18    <hr>↓
19    <div class="red">こっち</div>↓
20    <div class="red">は</div>↓
21    <div class="red">redクラス</div>↓
22    <div class="blue">はぐれblueクラス</div>↓
23  </body>↓
24 </html>↓
25 [EOF]

```

いままではidを使って特定の一つのHTMLタグを取り出して、そのスタイルを変えたりしてきた。ここでは複数のHTMLタグをまとめてスタイル指定する方法を学ぶ。

5~7行目はblueクラスのスタイル指定である。idは#id名で指定したが、classは.class名で指定する。

8~10行目はredクラスのスタイル指定である。

14~22行目を見ると、class="blue"とされたdivと、class="red"とされたdivがあることがわかる。前者がblueクラス、後者がredクラスに分類される。

3-7 getElementsByClassName

ポイント

・ getElementById は id 名を指定して特定のタグを選択したが、class 名を指定して、その class に属するタグ全てを選ぶ方法が getElementsByClassName である。

```
var a = document.getElementsByClassName("blue");
```

・ Elements が複数形になっていることからわかるように、この命令は複数の要素を得る。これらは配列という形式で変数に格納される。

・ a が配列の時、a の要素数は a.length, a の各要素は a[0], a[1], … a[a.length-1] である。これは詳しくは次の章でやろう。

課題

・ 次の HTML を書こう。

```

1 <html>↓
2   <head>↓
3     <title>get</title>↓
4     <script>↓
5       function sum(){↓
6         var inputs = document.getElementsByClassName("i");↓
7         var result = document.getElementById("sum");↓
8         var s = 0;↓
9         for(var i=0; i<inputs.length; i++){↓
10          s += parseFloat(inputs[i].value);↓
11        }↓
12        result.value = s;↓
13      }↓
14    </script>↓
15  </head>↓
16  <body>↓
17    <input class="i"> <br>↓
18    <input class="i"> <br>↓
19    <input class="i"> <br>↓
20    <input class="i"> <br>↓
21    <input class="i"> <br>↓
22    <input type="button" value="合計" onclick="sum()"> <br>↓
23    <input id="sum"> <br>↓
24  </body>↓
25 </html>↓
26 [EOF]
```

このプログラムでは、class が "i" になっているフォーム全ての値を合計している。22, 23 行目の <input> タグは、class が "i" でないので合計には含まれない。

3-8 配列

ポイント

- ・ getElementsByClassName の結果は配列である。
- ・ 配列とは、たくさんの変数をまとめたものである。
- ・ 配列は var と new Array を使って作る。

```
var a = new Array(10);          (この 10 は配列に含まれる変数の数)
```

このとき、a[0], a[1], … a[9] の 10 個の変数が作られる。この[x]の部分を実列の添字という。

- ・ 配列は for 文と相性が良い。

```
var sum=0;
for(var i=0; i<10; i++){
    sum = sum + a[i];
}
```

これを実行すると、i が 0, 1, …, 9 と動きながら sum=sum+a[i] が実行されて、最終的に sum が a[0] から a[9] までの合計になる。これが 3-7 の課題のプログラムでやっていたことである。

・ 下の例は大したことはしていないが、配列の用意、配列へのデータ代入、配列からのデータの取り出しの方法を示している。

```

1  <html>↓
2  <head>↓
3      <title>配列</title>↓
4      <script>↓
5          function calc(){↓
6              var a = new Array(100);↓
7              for(var i=0; i<100; i++){↓
8                  a[i] = i*i;↓
9              }↓
10             ↓
11             var d = document.getElementById("div1");↓
12             for(var i=0; i<100; i++){↓
13                 d.innerHTML += a[i] + "<br>";↓
14             }↓
15         }↓
16     </script>↓
17 </head>↓
18 <body>↓
19     <input type="button" value="start" onclick="calc()"> <br>↓
20     <div id="div1"></div>↓
21 </body>↓
22 </html>↓
23 [EOF]
```

3-9 練習問題 2

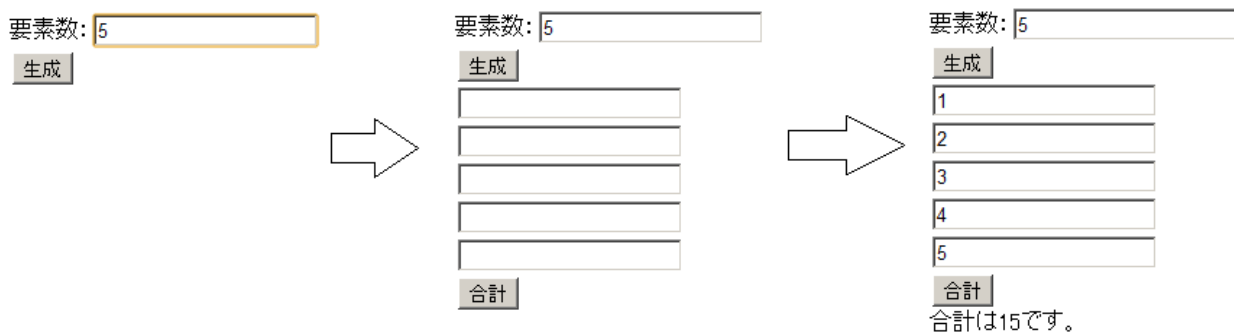
・ 3-3 と 3-7 の融合

次のようなプログラムを作れ。

まず要素数を入力し、生成を押すとフォームがその数だけと、合計ボタンがあらわれる。

次に、あらわれたフォームに数を入力し、合計ボタンを押すと、その合計が表示される。

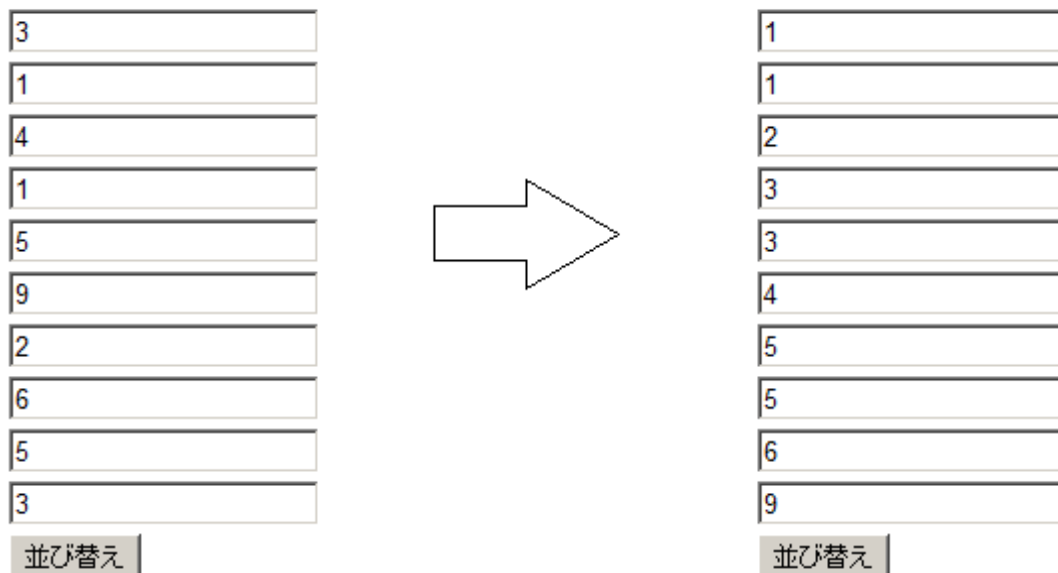
注意：””の中に、””をつかった文を含めたいときは、内側の””を"で記述する。



・ 並び替え

次のようなプログラムを作れ。

10 個の数字をフォームに入力し、並び替えボタンを押すと、小さい順に並び変わる。



3-10 乱数

ポイント

- ・ `Math.random()` と書くと、その部分は 0~1.0 のランダムな小数になる。
- ・ たとえば 0~n までのランダムな整数が欲しければ、
 `Math.floor(Math.random()*n)`
 と書けばいい（`Math.floor` は切り捨ての意味）

課題

- ・ ボタンを押したらそれぞれ 1/5 の確率で大凶、凶、吉、中吉、大吉が表示されるおみくじを作ろう。

3-11 応用

- ・雪

雪が上から降ってくるアニメーションをつくろう。

- ・雪の初期位置はランダム
- ・雪はランダムに左右に動きつつ落ちてくる
- ・下まで行った雪は上に戻る
- ・雪は全部同じ class
- ・アニメーションは setTimeout を使う

4日目 グラフィックス

1. 骨組みを作る
2. 図形の描き方
3. 色
4. マウスとの相互作用
5. 透明度
6. `for` 文による描画
7. 二重ループによる描画
8. 乱数による描画
9. アニメーション
10. 配列を使ったアニメーション
11. ゲーム

4-1 骨組みを作る

今日は processing.js というグラフィックスライブラリ（画像処理のための便利機能集みたいな）を使って、HTML+javascript のグラフィックプログラミングをしよう。

今日作るプログラムは共通の部分が多いので、先に共通する部分だけ作ってしまって、あとはこれをコピーして使いまわすことにする。早速次のプログラムを作ろう。（4_1_temp.html）

```

1 <html>↓
2 <head>↓
3 <title>テンプレート</title>↓
4 <script src="processing.js"></script>↓
5 <script>↓
6   function sketchProc(p){↓
7     p.setup = function(){↓
8       p.size(500, 500);↓
9       p.background(255, 255, 255);↓
10      p.smooth();↓
11    }↓
12    p.draw = function(){↓
13      p.background(255, 255, 255);↓
14      ↓
15      p.ellipse(250, 250, 100, 100);↓
16    }↓
17  }↓
18  function add(){↓
19    var c = document.getElementById("c");↓
20    p = new Processing(c, sketchProc);↓
21  }↓
22 </script>↓
23 </head>↓
24 <body onload="add()">↓
25   <div>↓
26     <canvas id="c" style="border:solid 1px black"></canvas><br>↓
27   </div>↓
28 </body>↓
29 </html>↓
30 [EOF]

```

実行して、真ん中に小さな円が表示されたら OK である。

- ・非常にざっくりとした説明

26 行目の<canvas>が、キャンバス（描画領域）を作っている。

4, 6, 18-21 行目 … processing.js を使うための準備。

7-16 行目 … 描画処理

7-11 行目 … 一番最初に一回だけ実行される命令。主にキャンバスの設定。

12-16 行目 … 一秒間に 60 回繰り返し実行される命令。

13 行目 p.background(r, g, b) … キャンバス全体を (r, g, b) の色で塗る

15 行目 p.ellipse(x, y, w, h) … (x, y) を中心とする、幅 w、高さ h の楕円を描く

4-2 図形の描き方

図形を描くには次の命令を使う。ただし、キャンバス左端から X、上から Y の点を (X, Y) と書く。

`p.point(X, Y)` 命令

(X, Y) に点を書く

`p.line(X0, Y0, X1, Y1)`

(X0, Y0) から (X1, Y1) に線を引く。

`p.rect(X, Y, W, H)`

(X, Y) を左上の頂点とした、横 W、高さ H の長方形を描く。

`p.ellipse (X, Y, W, H)`

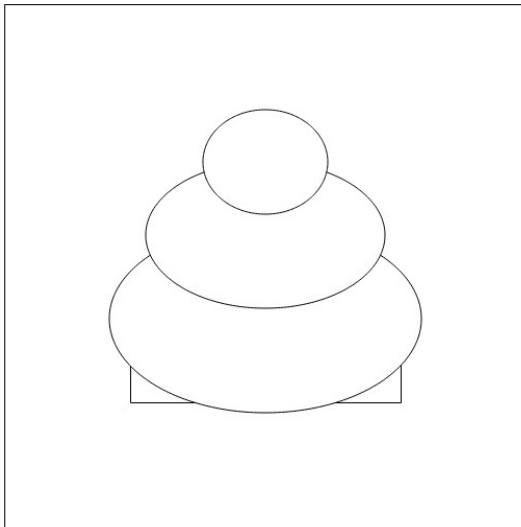
(X, Y) を中心とした、横 W、高さ H のだ円を描く。

これらの命令は、後に書いたほうが、上に重ねて描かれる。

課題

・今作ったテンプレートの 15 行目の次の行に、違う図形を描く命令を書き足して、図形がちゃんと表示されるか確認せよ。

・もうすぐ 2012 年ということで鏡餅でも描いてみましょう。



4-3 色

色の指定は次の4つの命令で行う。りんかく線と内部の塗りつぶしを別々に決める。

p.stroke(R, G, B)・・・りんかく線の色を R,G,B で指定

p.fill(R, G, B)・・・塗りつぶしの色を R,G,B で指定

p.noStroke()・・・りんかくなし

p.noFill()・・・塗りつぶしなし(りんかくだけになる)

R,G,B は 0~255 の値をとる。1 日目にも書いたがここ

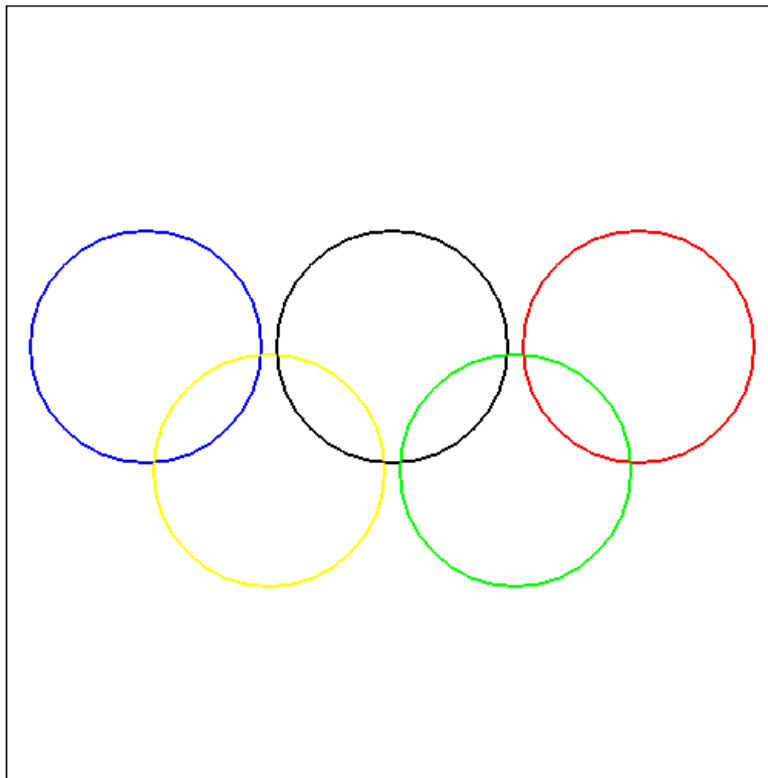
<http://www.htmq.com/color/colormame.shtml> を参考にするといい。

これらの命令は、それ以降の命令全てに影響することに注意しよう。

例えば、p.stroke(255, 0, 0);を実行すると、次に p.stroke か p.noStroke 命令が実行されるまで、描かれる図形のりんかく線は全て赤色になる。

課題

- ・オリンピックマークを描こう。



4-4 マウスとの相互作用

processing.js を使うと、マウスの位置や、マウスクリックの動作を手軽に決めることができる。

マウスの位置は、

```
p.mouseX, p.mouseY
```

で取得する。

マウスがクリックされた時の動作は、

```
p.mousePressed = function(){  
    ...  
}
```

と記述すれば良い。

課題

- ・ テンプレートの 15 行目を、

```
p.ellipse(p.mouseX, p.mouseY, 20, 20);
```

と変えてみよう。p.mouseX, p.mouseY には、マウスの X 座標（左からの距離）と Y 座標が自動的に入る。

テンプレートの 12-16 行目、これは p.draw という命令を作っていて、processing.js ではこの名前の命令は自動的に一秒間に 60 回呼ばれる仕組みになっている。

ここでは、呼び出されるたびに、今まで書いてある絵を消して（13 行目）、マウスの位置に円を描くので、円がマウスについてきているように見える。

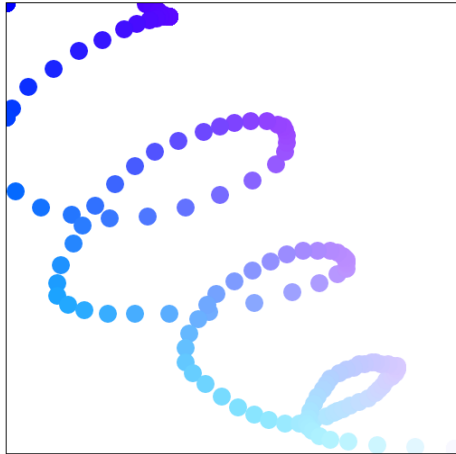
- ・ では、13 行目の p.background(255, 255, 255) 命令（キャンバス全体を白で塗りつぶす）を消してしまったらどうなるか？

- ・ 16 行目と 17 行目の間に、次のマウスがクリックされた時の動作を入れてみよう。

```
p.mousePressed = function(){  
    p.ellipse(p.mouseX, p.mouseY, 100, 100);  
}
```

課題

1. マウスがクリックされたら画面全体が白く塗りつぶされて消えるようにしよう。
2. 次にマウスの X 座標、Y 座標によって違う塗りつぶしの色の円が描かれるようにしよう。
(例)



3. 2日目の2-8を参考に塗りつぶしの色をフォームから取得するようにしよう。
4. クリックしたら白く塗りつぶすのではなく、クリックすると円を描くモードになって、もう一度ク



赤	0
緑	255
青	255

リックすると何も描かないモードに切り替わるようにせよ。

4-5 透明度

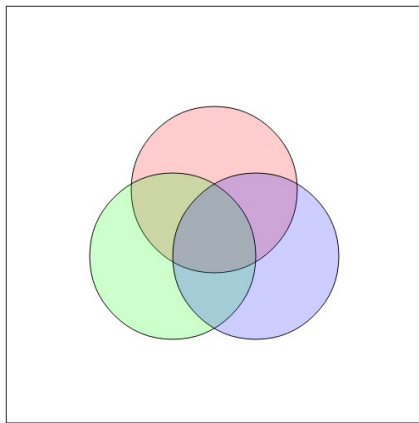
今まで色は R, G, B の 3 色で指定してきたが、R, G, B, A の 4 色で指定することも出来る。A は透明度を表し、0 が完全な透明、255 が不透明である。たとえば、

```
p.fill(255, 0, 0, 100);
```

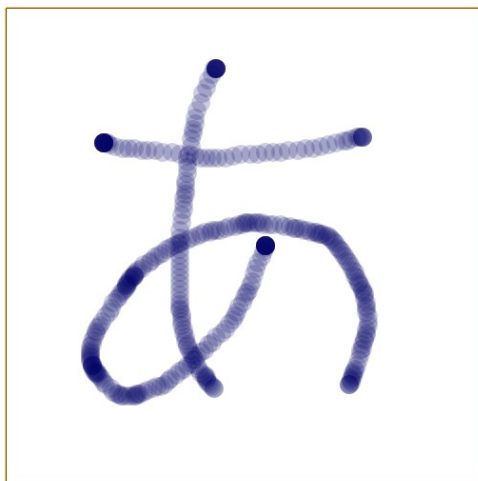
と書けば、赤の半透明な塗りつぶしが出来る。

課題

- ・ 次の絵を描け。



- ・ さっきのに透明度も組み込むと、色の濃淡が出せるようになる。



赤	0
緑	0
青	100
透明度	50

4-6 for 文による描画

似たような図形をたくさん描きたいときは for 文を使う。

昨日 3-3 まで行かなかった人は 3-3 を見よ。とくに例と書いてあるところを見ておこう（プログラムは書き写さなくていい）。

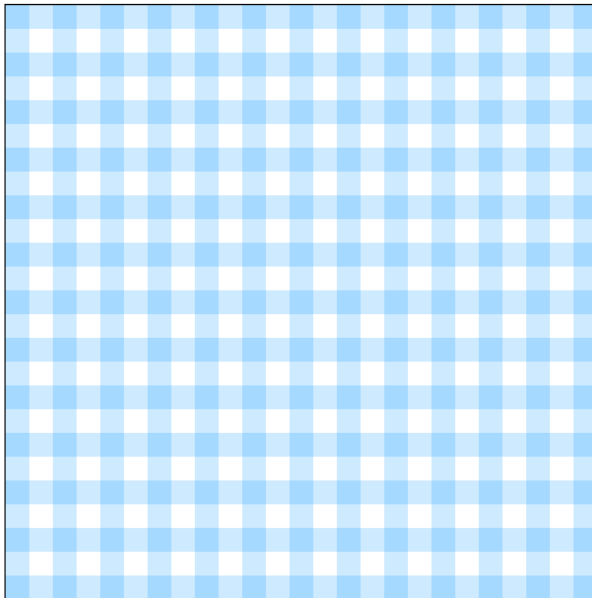
課題

- ・ テンプレートの 15 行目の代わりに、次の文を書こう。

```
for(var i=0; i<=500; i=i+10){  
    p.line(i, 0, i, 500);  
}
```

この for 文では、i は 0, 10, 20, ..., 500 と動く。だから、p.line(0, 0, 0, 500);, p.line(10, 0, 10, 500);, ... を連続して実行したのと同じ結果になる。

- ・ 次の絵を描くプログラムを作れ。（二重ループは使わないよ）



4-7 二重ループによる描画

for ループの中に for ループをいれると表のように何かを描くときに便利である。
昨日 3-4 まで行かなかった人は、先に 3-4 に目を通そう。

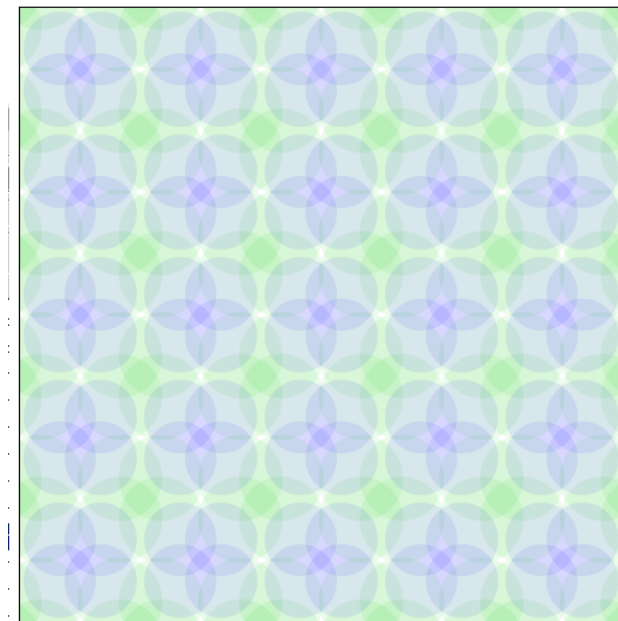
課題

- ・ テンプレートの 15 行目の代わりに、次の文を書こう。

```
for(var i=0; i<=500; i=i+20){  
    for(var j=0; j<=500; j+=20){  
        p.ellipse(i, j, 20, 20);  
    }  
}
```

ここでは、 (i, j) は $(0, 0), (0, 20), \dots, (0, 500), (20, 0), (20, 20), \dots, (20, 500), \dots, (500, 500)$ という順序で動いていく。 (i, j) は円の中心になるから、マス目状に円が描かれる。

- ・ 次のような絵を描こう。



4-8 乱数による描画

自然にあるものはたいていランダムさが入っている。ここでは「乱数」を使ってランダムさを表現することを学ぶ。

乱数とは、サイコロのようなもので、やるたびに結果が変わる命令である。

```
p.random(0, 100);
```

こう書くと、これは 0~100 までのランダムな値になる。

```
p.random(-5, 5);
```

のように負の数が入っても良い。これは -5~5 のランダムな値になる。

課題

・テンプレートの 12-16 行目を次のように書こう。

```
p.draw = function(){  
  p.background(0, 0, 0);  
  p.noStroke();  
  p.fill(255, 255, 255);  
  for(var i=0; i<=500; i=i+1){  
    var d = p.random(0, 3);  
    p.ellipse(p.random(0, 500), p.random(0, 500), d, d);  
  }  
}
```

このプログラムでは、黒い背景に 500 個の小さい円を描く。円の位置は、左から 0~500 のランダムな位置、上からも 0~500 のランダムな位置で、円の直径は、0~3 のランダムな値が入る。

なぜ直径は p.ellipse のなかで rand(0, 3) と書かないのかというと、そうすると横幅と縦幅が違う値になって楕円を描いてしまうからである。変数 d を介することで、横幅と縦幅を同じにして円にしている。

p.draw は 1 秒間に 60 回呼ばれ、ここでは呼ばれるたびに画面を真っ黒にして (p.background(0, 0, 0) による)、円を描き直しているため、電波の入っていないテレビのような映像が表示される。

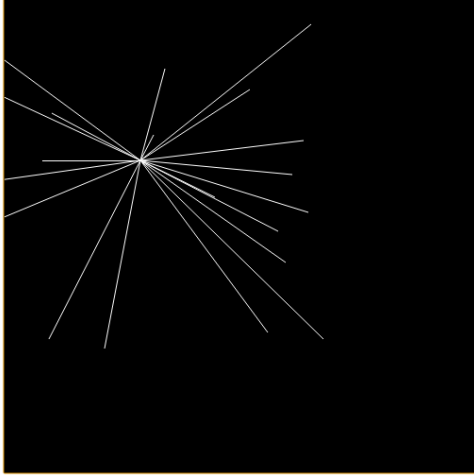
・このプログラムの 10 行目と 11 行目の間に、

```
p.noLoop();
```

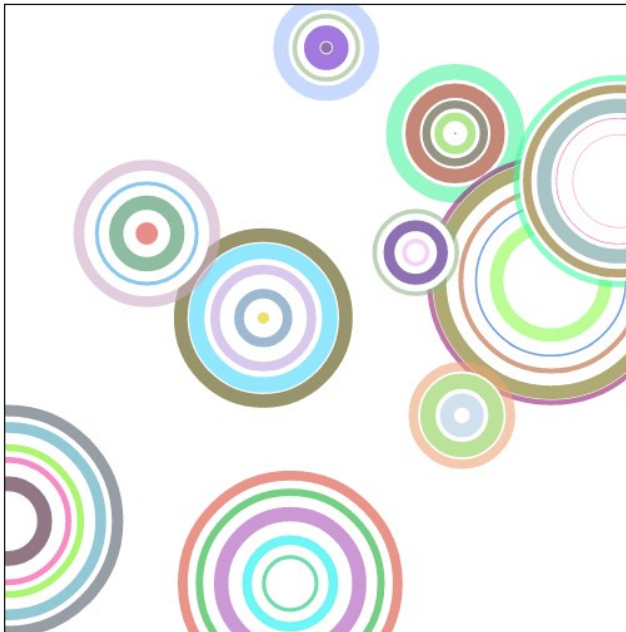
と書いてみよう。こうすると、p.draw が一回だけしか呼ばれなくなるので、映像は静止する。星空のような絵が見えるはずである。

- ・ 次のようなプログラムを作れ。

真っ黒な背景で、マウスをクリックすると、クリックした点を中心に、ランダムな方向に、ランダムな長さの白い線が 20 本引かれる。こうするとクリックしたところから画面にヒビが入ったように見える？



- ・ こんな絵も描ける。（色のついた円と真っ白な円を、同じ位置に、直径を小さくしながら交互に描いている。これをランダムな位置に 10 箇所表示している。）



4-9 アニメーション

描かれる図形の位置などを、p.draw が呼び出されるたびにちょっとずつ変えていけば、動いているように見える。これを実現するには、図形の変数で記録しておき、p.draw が呼ばれるたびにその変数の値を少しずつ変えていけばいい。

課題

- ・テンプレートの 12-16 行目を次のように書きなおそう。

```

12 |         var x = 10;↓
13 |         var y = 10;↓
14 |         var vx = 2;↓
15 |         var vy = 1;↓
16 |         p.draw = function(){↓
17 |             p.noStroke();↓
18 |             p.fill(255, 255, 255);↓
19 |             p.rect(0, 0, 500, 500);↓
20 |             p.stroke(0, 0, 0);↓
21 |             p.ellipse(x, y, 20, 20);↓
22 |             x = x + vx;↓
23 |             y = y + vy;↓
24 |             if(x > 500) vx = -vx;↓
25 |         }↓
26 |     }↓

```

まず、12-15 行目は、表示する円の中心の位置(x, y)と、x 方向速度 vx, y 方向速度 vy を記録する変数を作っている。これらの変数の値は、draw 命令が終わっても保存され、次に draw 命令が呼ばれたときに、この保存しておいた値を使って円を描きなおす。

17-19 行目では、真っ白な四角形で画面全体を塗りつぶしている。これは background(255, 255, 255)と同じ意味である。

22-23 行目では、位置を更新している。

24 行目では、位置がキャンバス右端を超えたら、x 方向速度を逆向きにしている。これによって、右端で跳ね返るようにみえる。

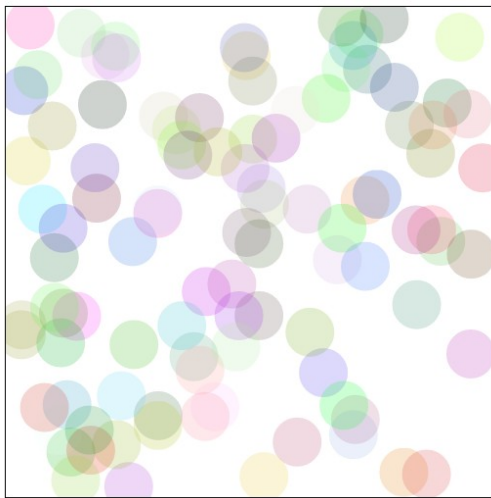
- ・12-15 行目の数字を変えて、何が起こるか試してみよう。
- ・4 方向全部跳ね返るようにしよう。
- ・18 行目を p.fill(255, 255, 255, 100)とすると何が起こるか？

4-10 配列を使ったアニメーション

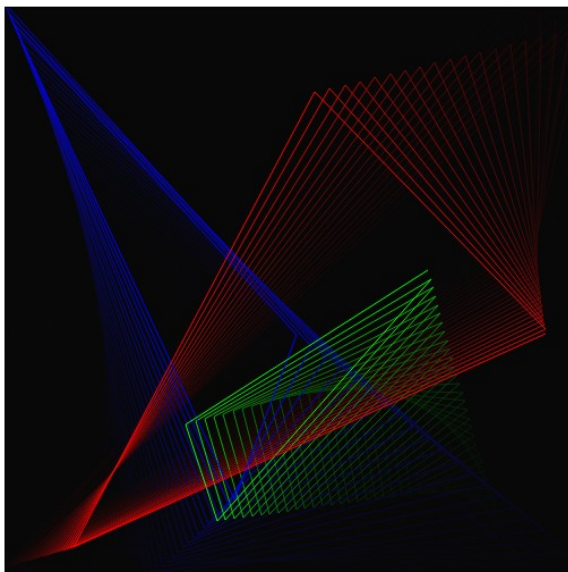
たくさんの変数をまとめて作りたいときは配列を使う。(3日目の3-8に目を通しておこう)
前のページの続きで、表示される円を1つだけじゃなくてたくさんにすることを考える。
このとき変数 x , y , vx , vy は表示する円の数だけ必要だから、配列を使うのが適している。

課題

- ・昨日 3-8 まで進んでいた人は、配列を使って表示される円の数を増やしてみよう。
進んでいなかった人は、次のページに書いてあるプログラムを理解して、書き写そう。
- ・円の色をランダムにしよう。ただし、1つの円に注目する限りでは、その円の色は時間がたっても変わってはいけない。



- ・いままで円の中心だったところを三角形の頂点にすると、次のようなラインアートが描ける。



(前頁の一つ目の課題の解答) テンプレートの 6-17 行目を次のように書く。

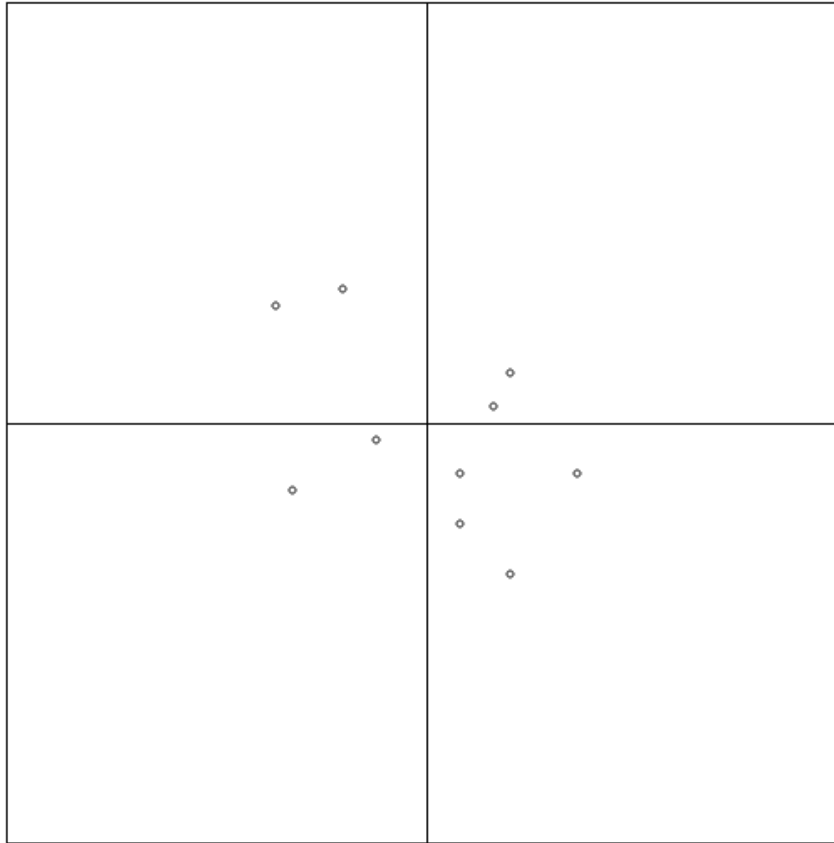
```
function sketchProc(p){↓
  var n;↓
  var x;↓
  var y;↓
  var vx;↓
  var vy;↓
  p.setup = function(){↓
    p.size(500, 500);↓
    p.background(255, 255, 255);↓
    p.smooth();↓
    n = 100;↓
    x = new Array(n);↓
    y = new Array(n);↓
    vx = new Array(n);↓
    vy = new Array(n);↓
    for(var i=0; i<n; i++){↓
      x[i] = p.random(10, 490);↓
      y[i] = p.random(10, 490);↓
      vx[i] = p.random(-5, 5);↓
      vy[i] = p.random(-5, 5);↓
    }↓
  }↓
  p.draw = function(){↓
    p.noStroke();↓
    p.fill(255, 255, 255);↓
    p.rect(0, 0, 500, 500);↓
    p.stroke(0, 0, 0);↓
    for(var i=0; i<n; i++){↓
      p.ellipse(x[i], y[i], 20, 20);↓
      if(x[i] > 490) vx[i] = -vx[i];↓
      if(y[i] > 490) vy[i] = -vy[i];↓
      if(x[i] < 10) vx[i] = -vx[i];↓
      if(y[i] < 10) vy[i] = -vy[i];↓
      x[i] = x[i] + vx[i];↓
      y[i] = y[i] + vy[i];↓
    }↓
  }↓
}
```

・配列の別の応用例（3-9 を終えた人向け）

画面をクリックすると、フォームに入力した x, y 座標の点をグラフにプロットする。

p.mousePressed の中で、document.getElementsByClassName を実行する。

x 座標たちと y 座標たちのフォームは別の class にしておくが良い。



x: -30	, y: 10
x: 40	, y: -10
x: 50	, y: 90
x: 20	, y: 60
x: 50	, y: -30
x: -50	, y: -80
x: -90	, y: -70
x: 90	, y: 30
x: 20	, y: 30
x: -80	, y: 40

4-11 ゲーム

4-10 で作ったプログラムを応用して、跳ねまわる円から避けるゲームを作ってみよう。

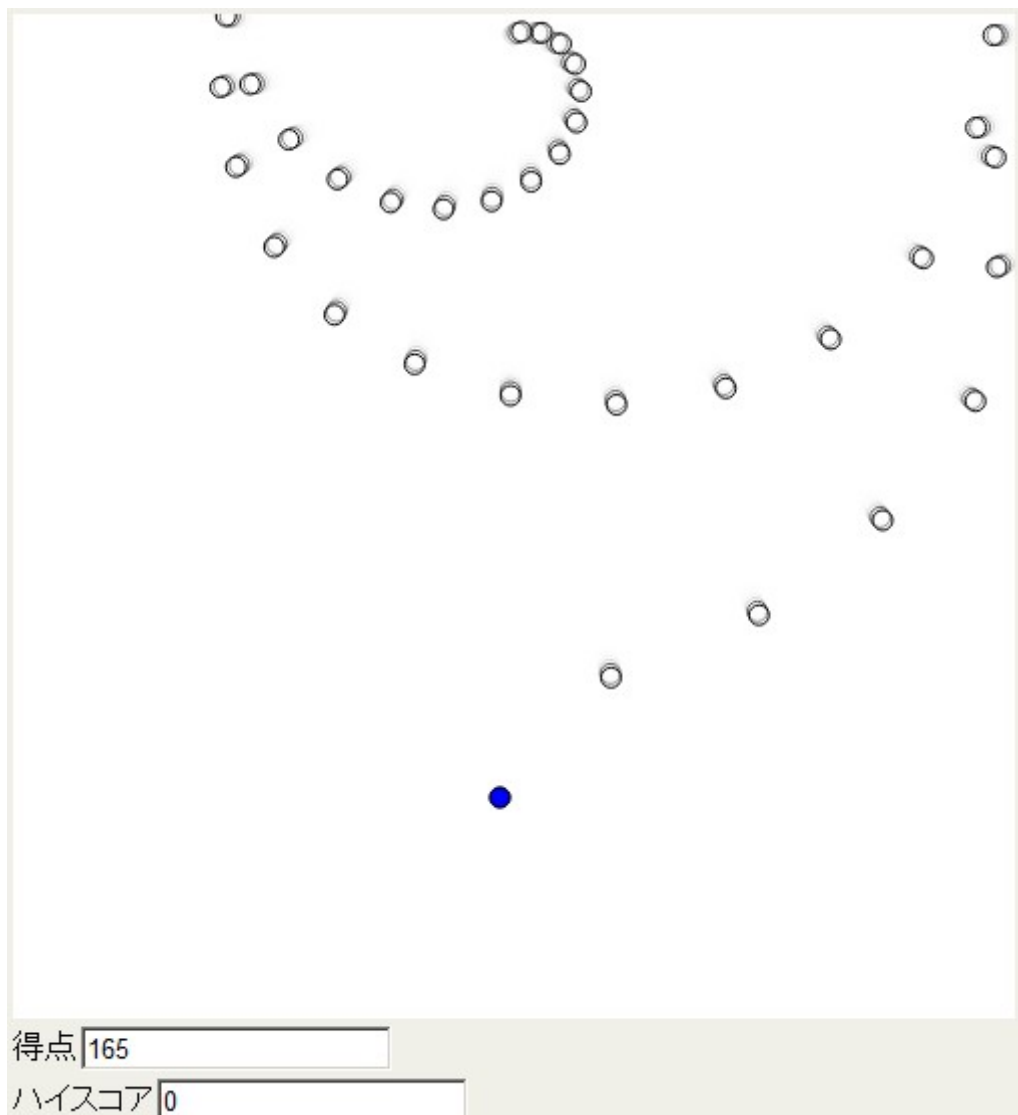
すべての円に対して、マウスとの距離を計算して、円の半径以内にマウスが入っていたらぶつかったとみなしてゲームオーバーにするだけである。

二点間の距離は

`p.dist(x1, y1, x2, y2)`

で計算できる。

時間のある人は得点やハイスコアをつけてみよう。



付録

- ・今回使用したソフトウェア
 - エディタ：Terapad
 - ブラウザ：Google Chrome
 - グラフィックライブラリ：processing.js（4日目のみ）いずれもフリー。検索すればすぐに出てきます。
- ・作ったHTMLをネットに公開するには
 1. レンタルサーバーを借りる。無料で借りられるところもたくさんある。
 2. サーバーに作ったファイルをアップロードする。個人情報の流出などには十分注意しよう。
- ・情報オリンピックに出るには
 - 次の予選は来年の12月。
 - まずは「アルゴリズム」と「データ構造」を勉強しよう。
 - 言語はC、C++を勉強しよう。今回やったjavascriptと大きくは変わらない。
 - 情報オリンピックのサイトに過去問がある。
- ・質問がある・授業で扱ったデータが欲しい
 - e.mattya@gmail.com
 - までメールを送って下さい。

4日間ありがとうございました(^_^)/~