

JavaScript 05

Generative Art

Daisy Nakamura



LandSkip CTO

ねろさん

技術って筆なんだよね

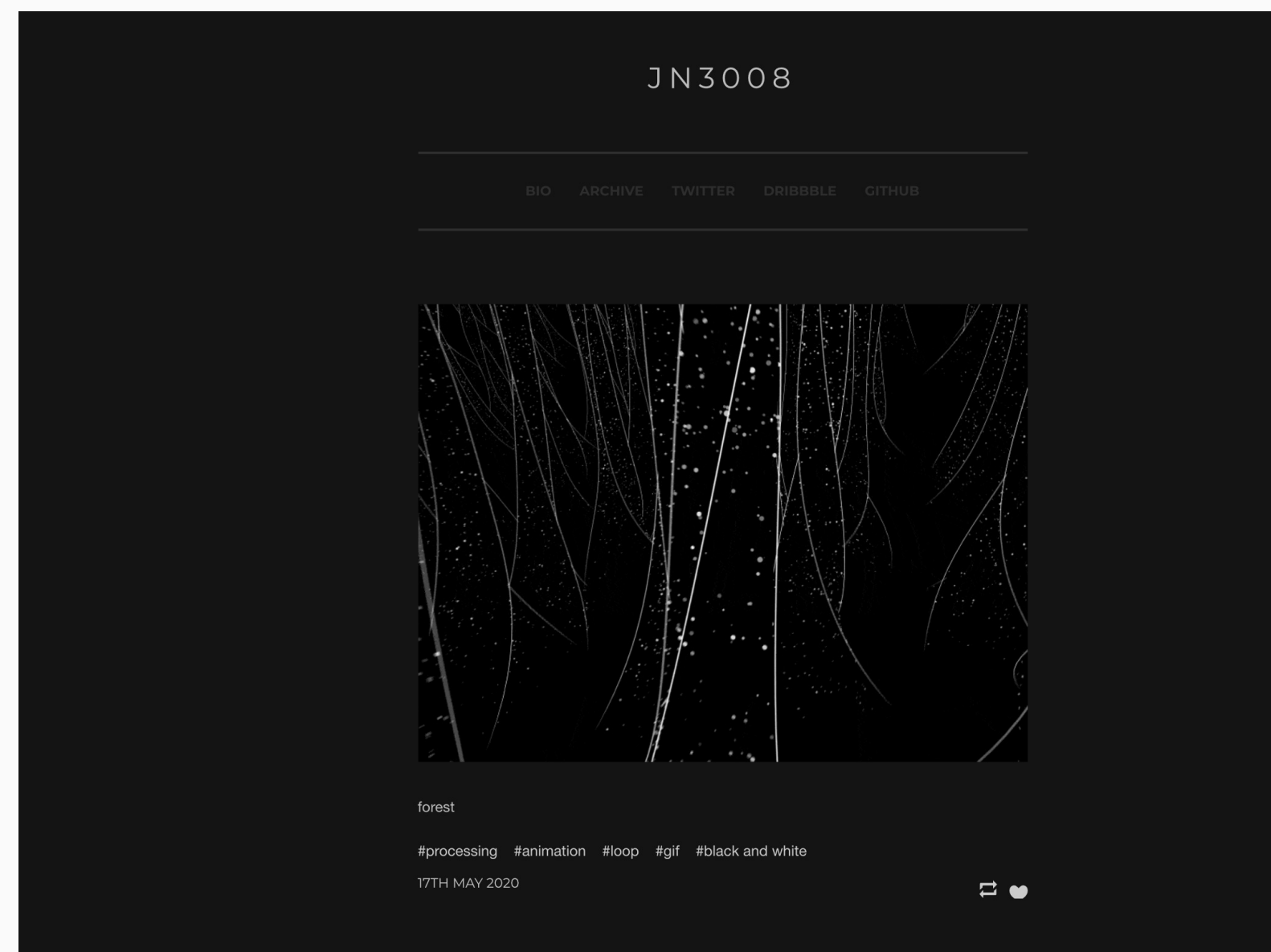
その筆で何を描くかが重要



Matt Pearson

ジェネラティブ・アートに共通言語はありません。その言語は日本語でも英語でもありません。
同時にそれは、Processing HTML5 あるいはC++でもありません。
ジェネラティブ・アートはアルゴリズムを土台にした不可知論的な構築物です。
そのルーツは自然や世界の動きを表す数学の中にあります。
ジェネラティブ・アートは文化の境界に目もくれません

ジェネラティブ・アート参考サイト



<https://jn3008.tumblr.com/>

今日のテーマ



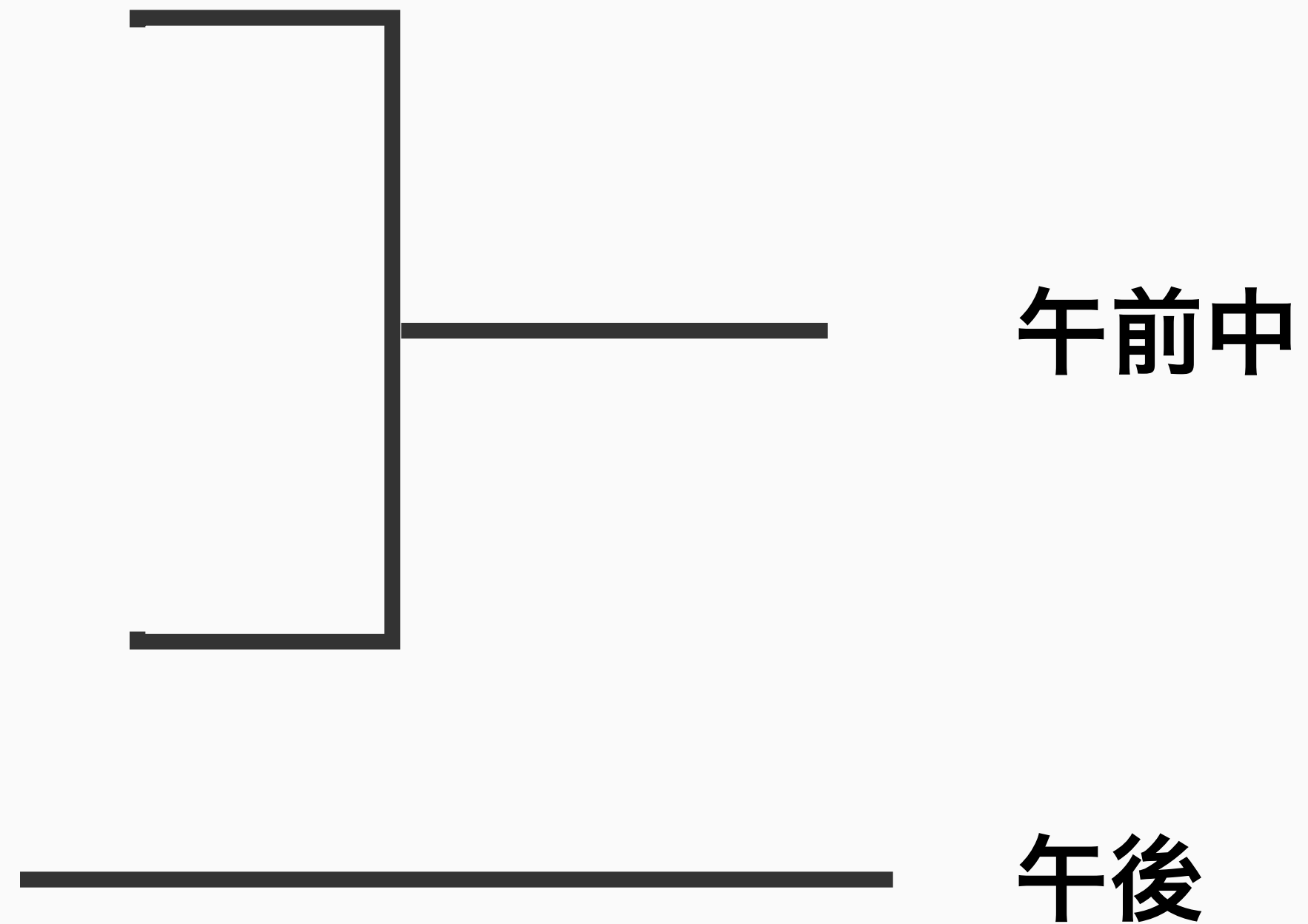
コードを書いてアートを描く

アジェンダ

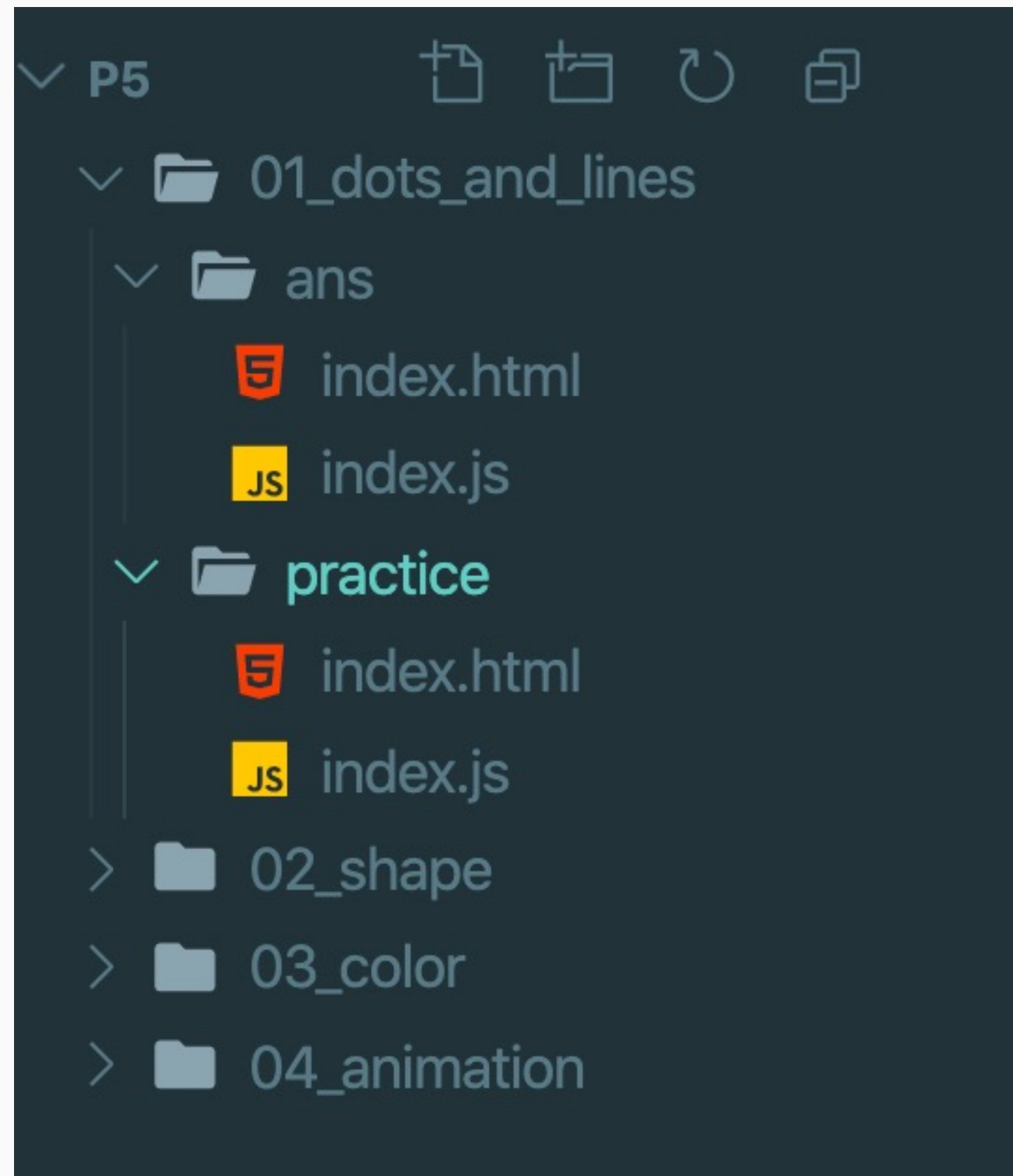


使用技術
p5.js

- 点と線
- 形
- 色
- 動き
- 演習



サンプルファイルの構成



資料のチャプター毎にフォルダ分けされてます。

チャプター内のフォルダの中にansとpracticeがあります

Practiceの中のindex.jsにコードは書いてください

授業の進め方

Study

Practice

各スライドにはタグがついてます

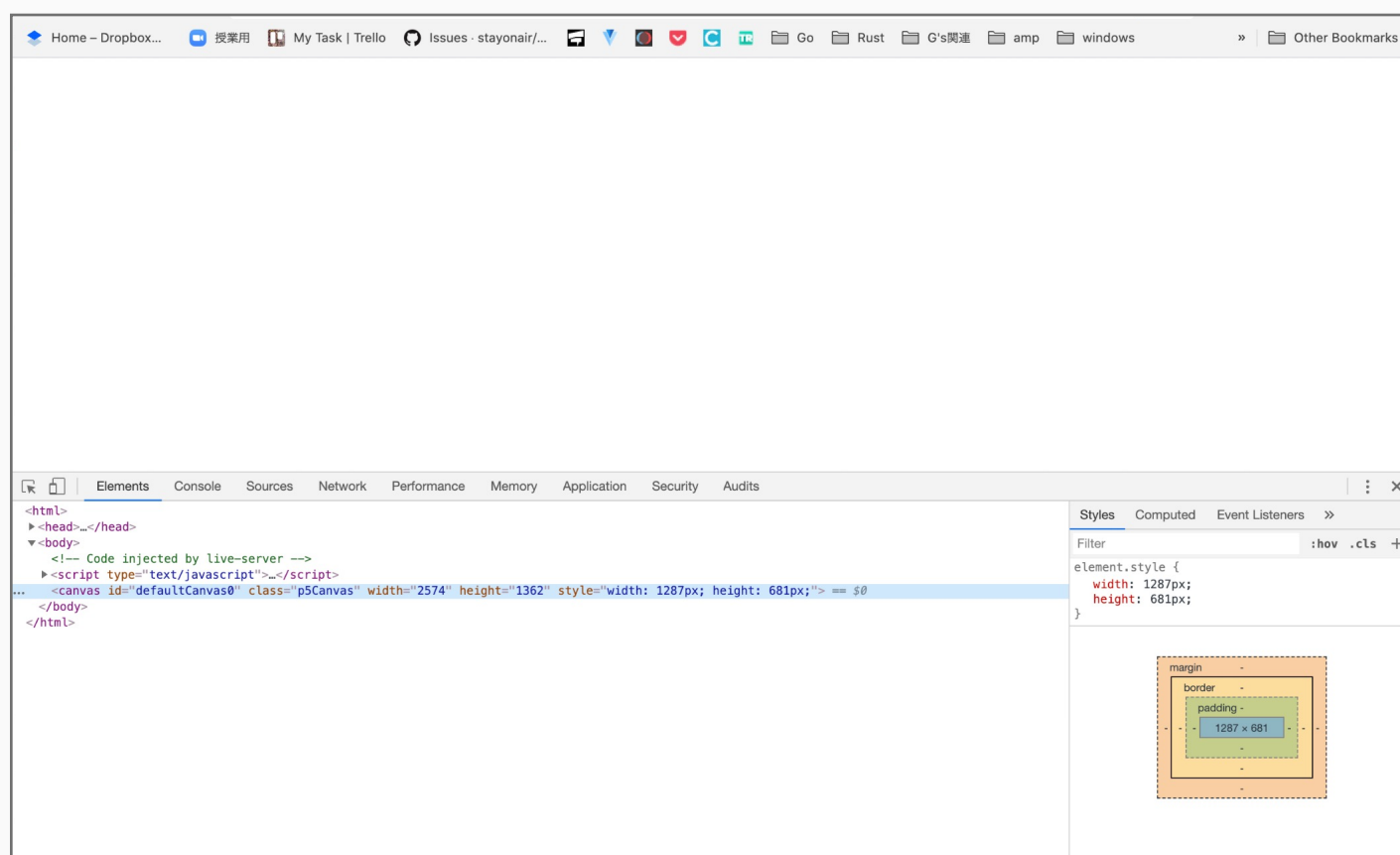
Practiceの時にになったら実際にコードを書いていただきます

点と線

dots and lines

createCanvas()

Study

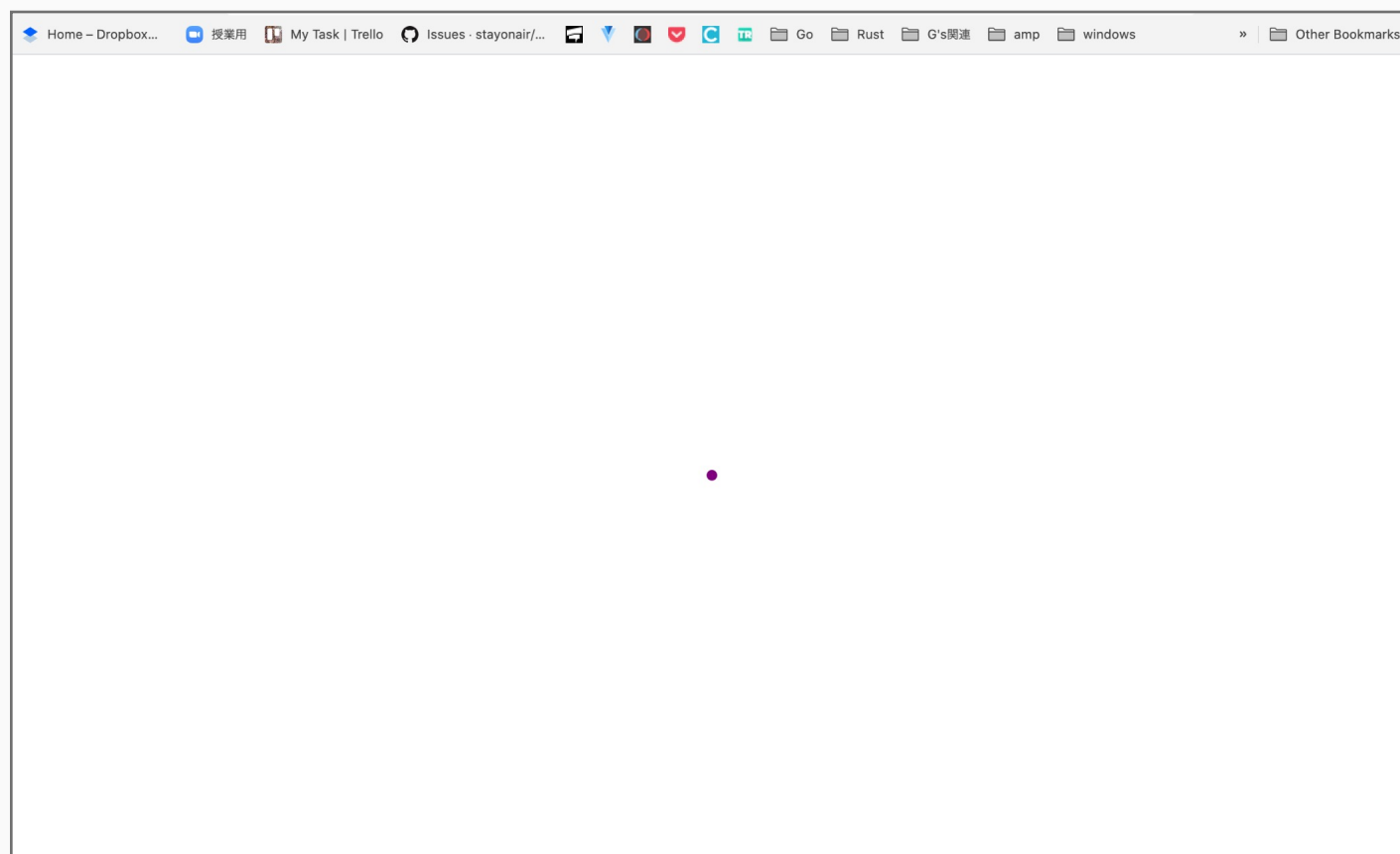


P5.jsではまず`createCanvas()`メソッドで描画領域を作る事から全てが始まります。
`createCanvas`メソッドは第一引数に横幅、第二引数に縦幅を指定することで
HTMLに`canvas`タグが挿入されます。

`createCanvas(400,600)`

point()

Study



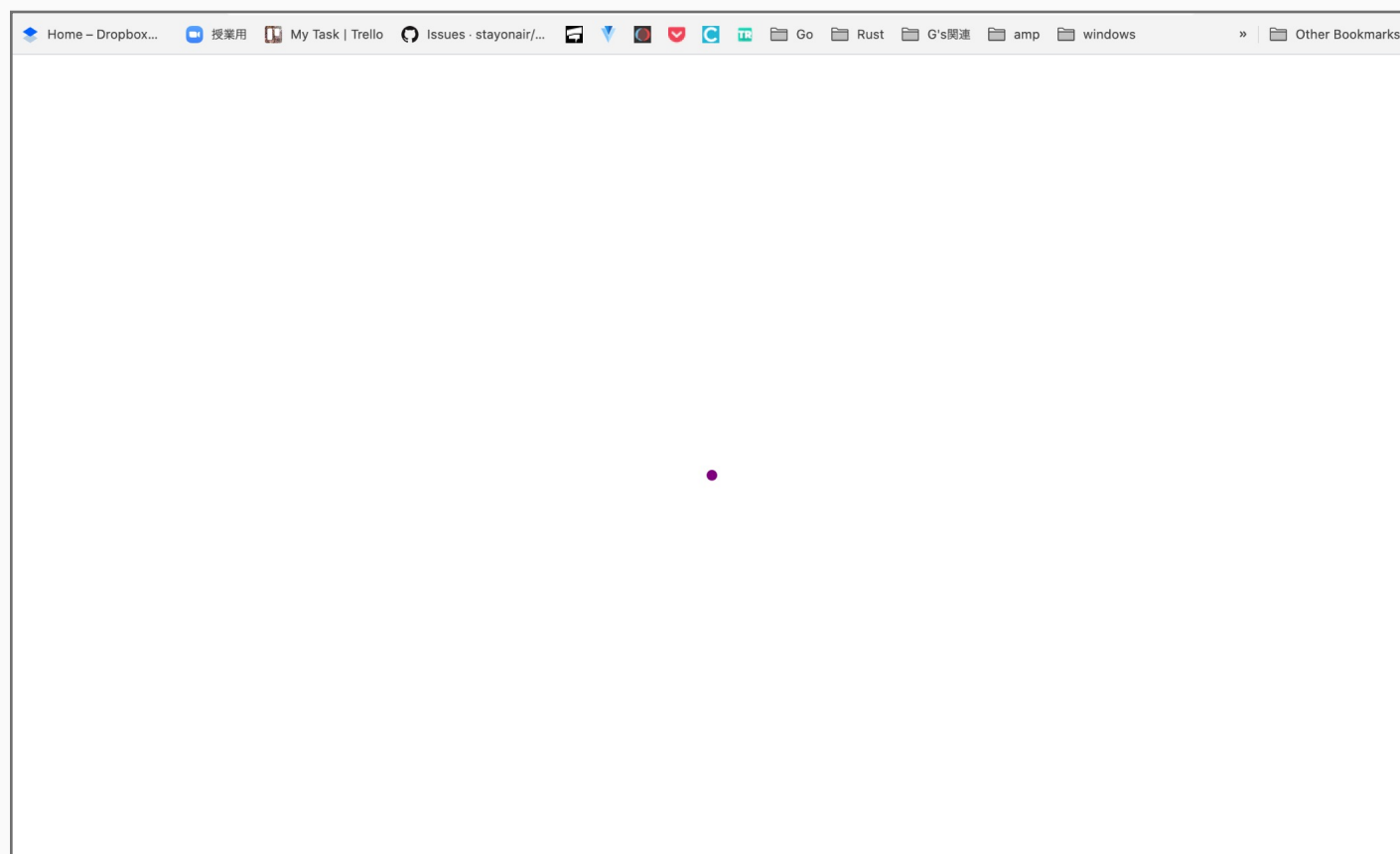
ありとあらゆるものは点の集合により表現されます。

点の描画こそがgenerative artにおけるHello Worldなのです。

点はpointメソッドにx座標とy座標（横の位置と縦の位置）を渡す事で描画されます

```
point(200,300)
```

stroke() strokeWeight()

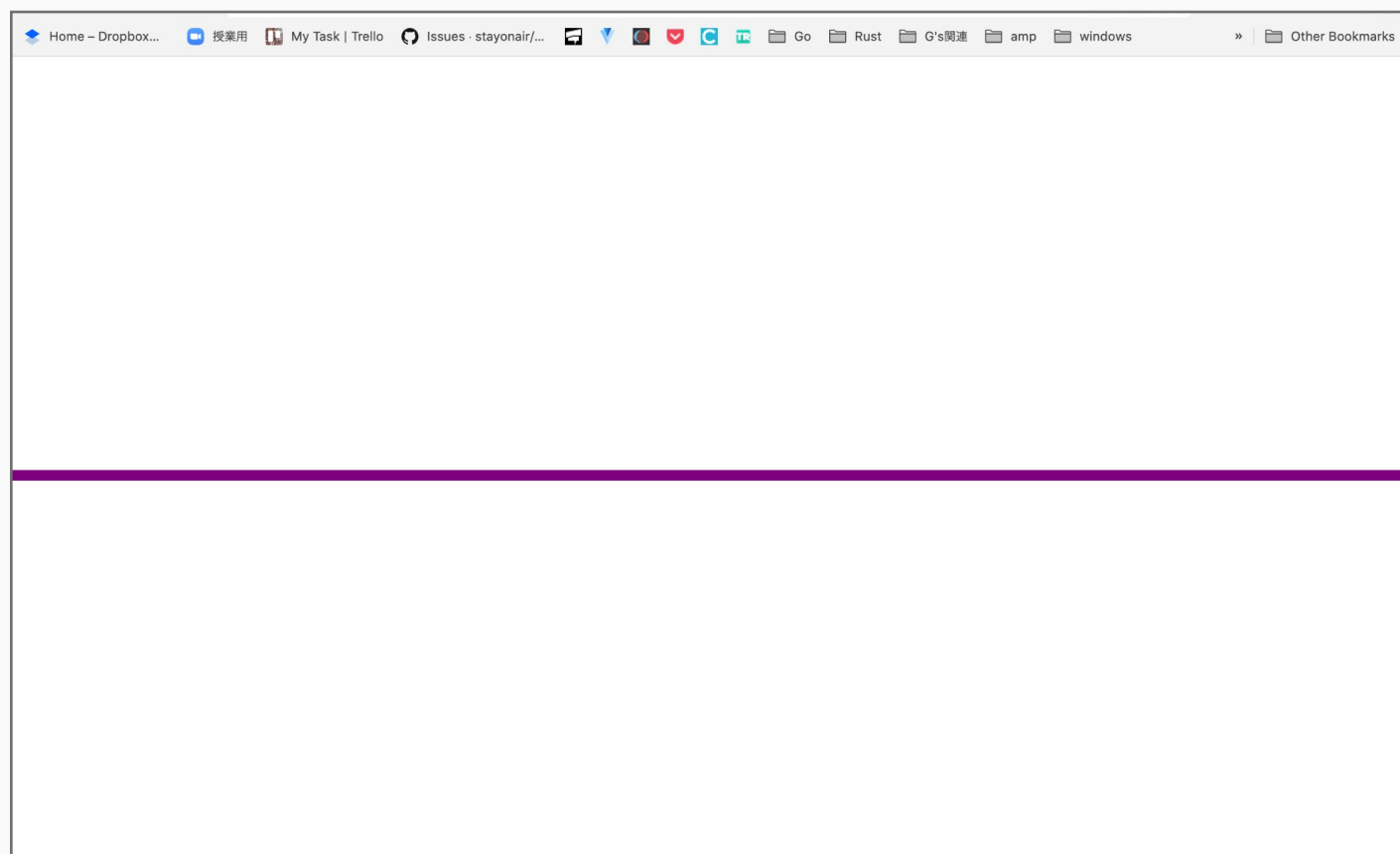


点の色はstrokeメソッド

点の太さはstrokeWeightメソッドで指定します

```
stroke("purple")  
strokeWeight(10)
```

線を描く間違った方法



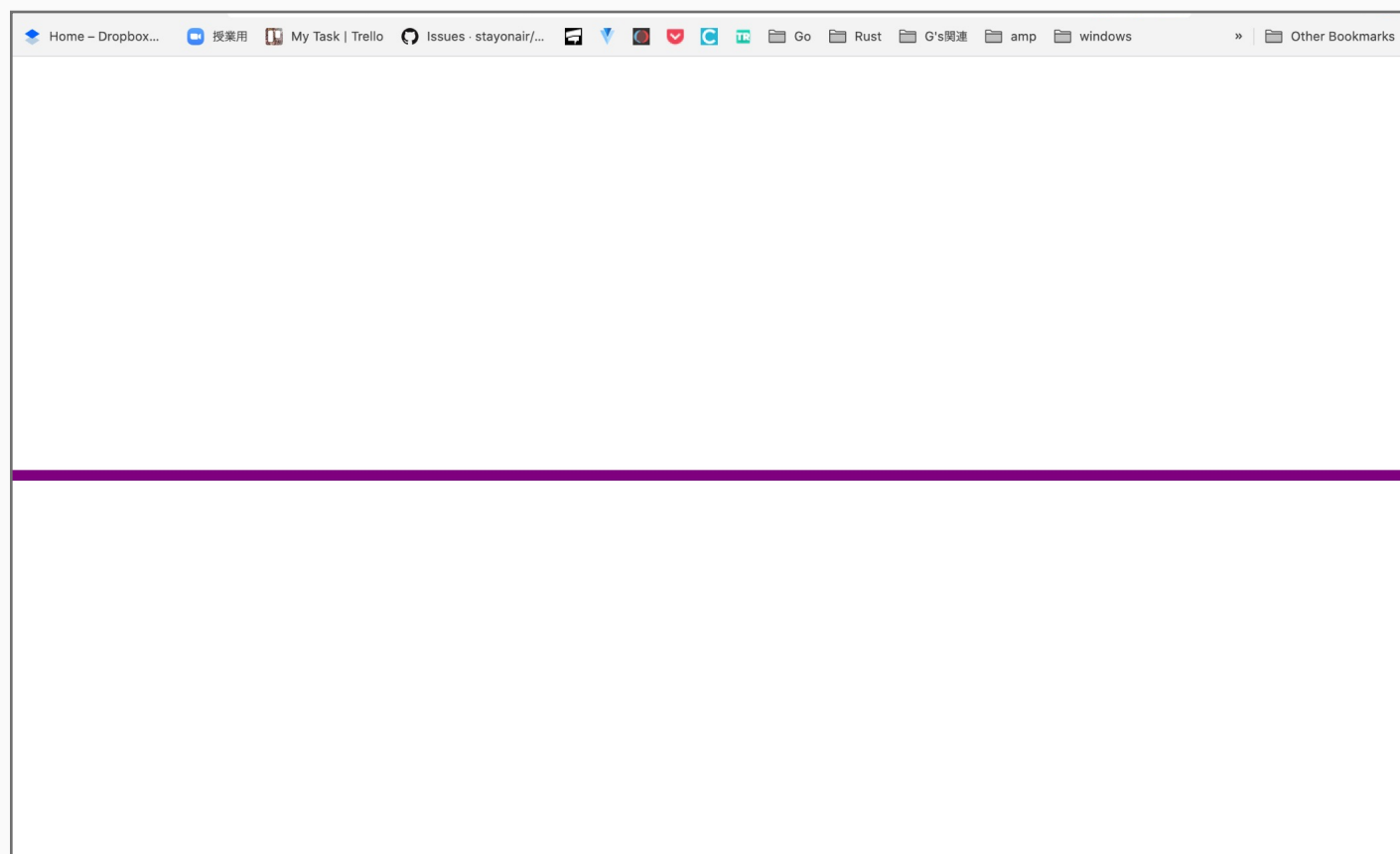
線は点の集合です。

繰り返し処理で少しずつ位置をずらせば線が描画出来ます？

```
for (let l = 0; l < width; l++) {  
  point(l, 300);  
}
```

line()

Study



線を引くための本当の方法はlineメソッドです。

lineメソッドには始点のx,y座標、終点のx,y座標を渡します。

p5.jsには線や図形を描くためのメソッドが用意されているので安心です。

でも忘れないでください。全ては点の連続、つまり座標が重要なのです。

```
line(0, 300, 200, 300);
```

Practice

```
01_dots_and_lines > practice > index.js > ...  
1  const width = window.innerWidth;  
2  const height = window.innerHeight;  
3  
4  const centerX = width / 2;  
5  const centerY = height / 2;  
6  
7  function setup() {}  
8  
9  function draw() {}  
10
```

ここまでの情報でHello Worldしてみましょう

01_dots_and_lines > practice > index.js

まとめ

dots and lines

- まずcreateCanvasする
- 描画は座標の指定によって行われる
- 横軸をx座標
- 縦軸をy座標と呼ぶ

形

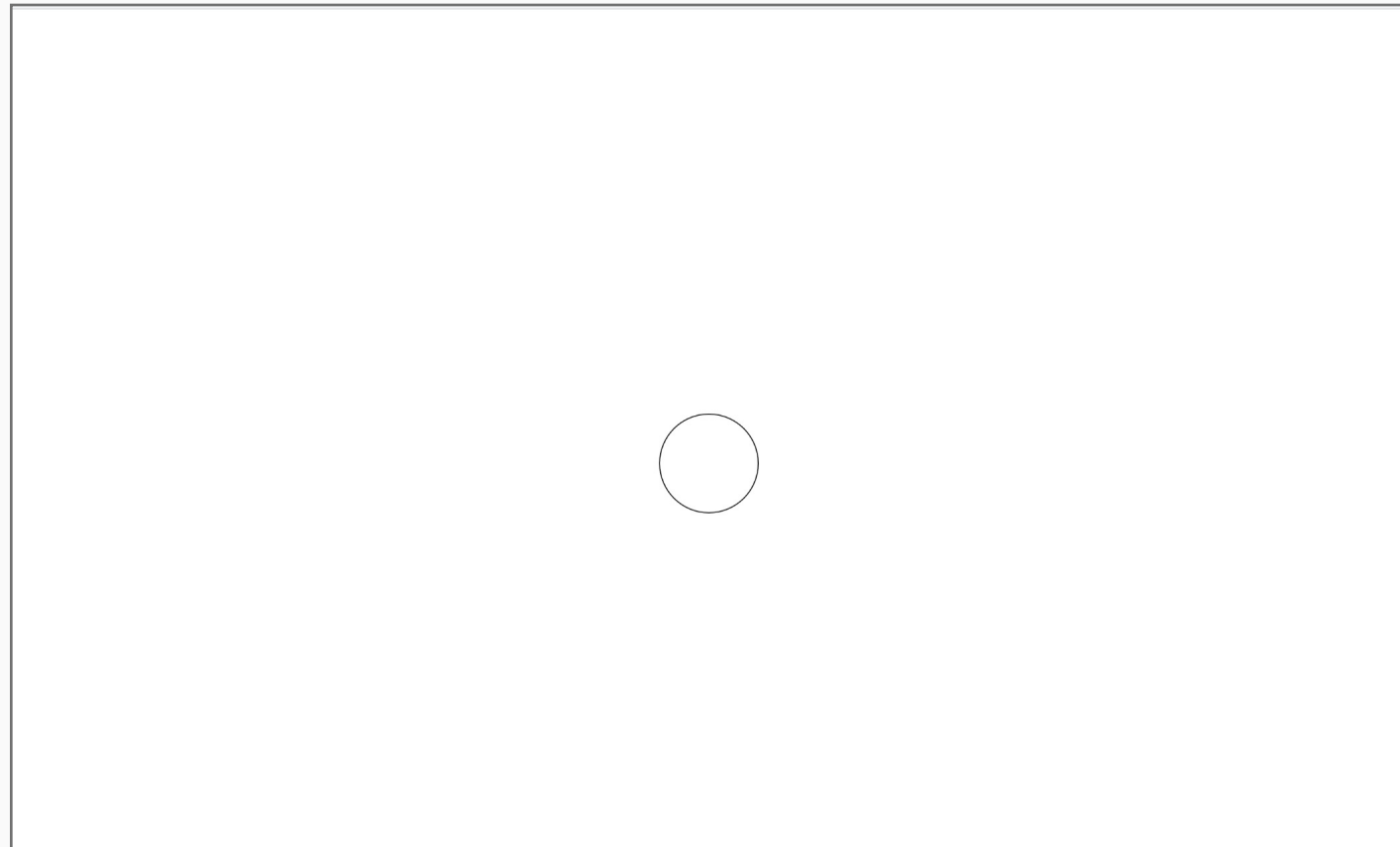
shape

形

shape

p5.jsにはlineメソッドのように、
形を描画するためのメソッドがいくつか用意されています。
しかし、本当に重要な事は角度についてです。
まず図形描画のメソッド、次に角度について説明します。

ellipse()



円の描画はellipseメソッドによって行います。引数が4つあります。

第一引数に中心のx座標

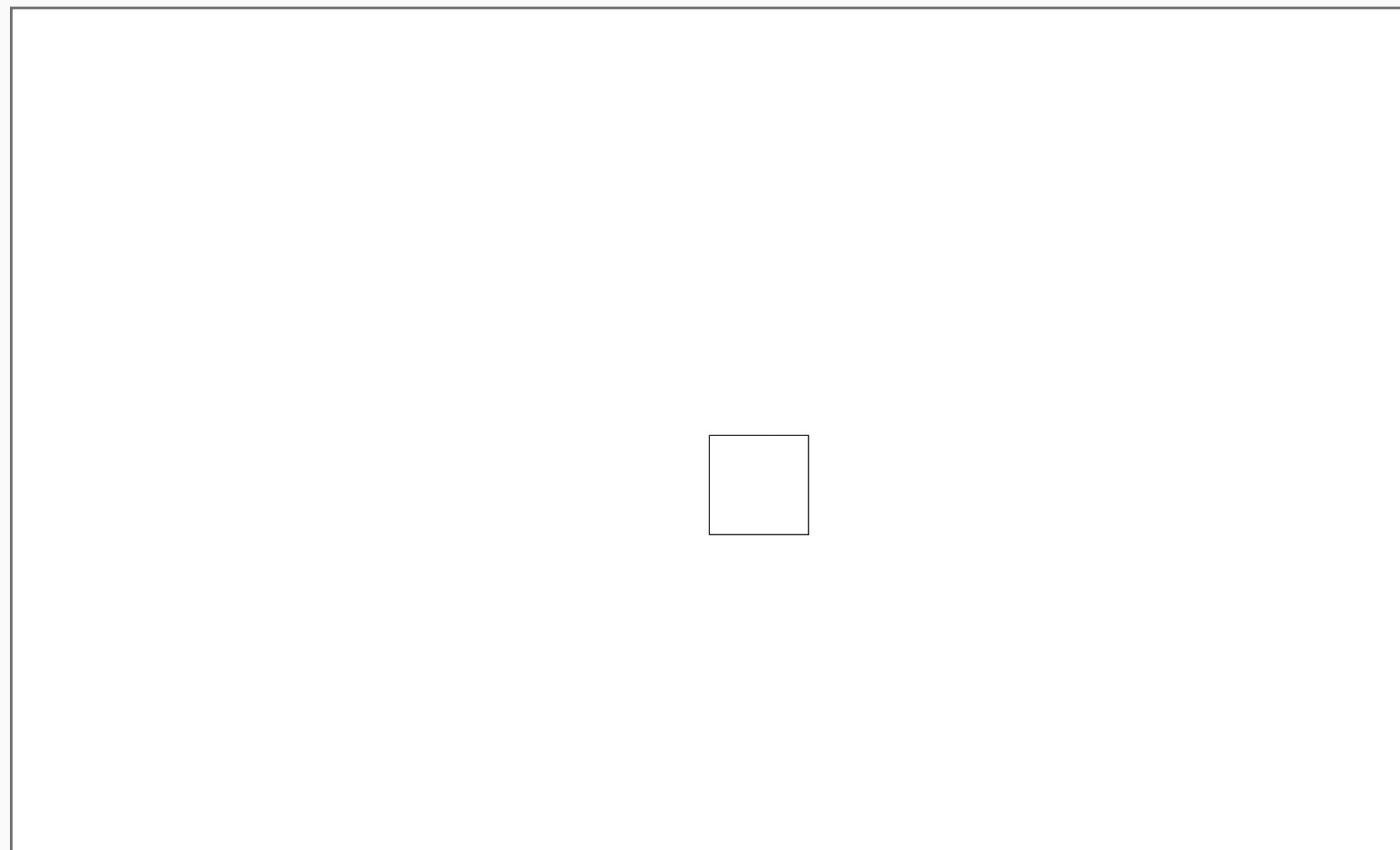
第二引数に中心のy座標

第3引数に横幅

第四引数に縦幅を指定します（省略可）

```
ellipse(200, 300, 10, 10);
```

rect()



四角形の描画はrectメソッドによって行います。引数はellipseと一緒にです。

第一引数に四角形の左端の頂点のx座標

第二引数に四角形の左端の頂点のy座標

第3引数に横幅

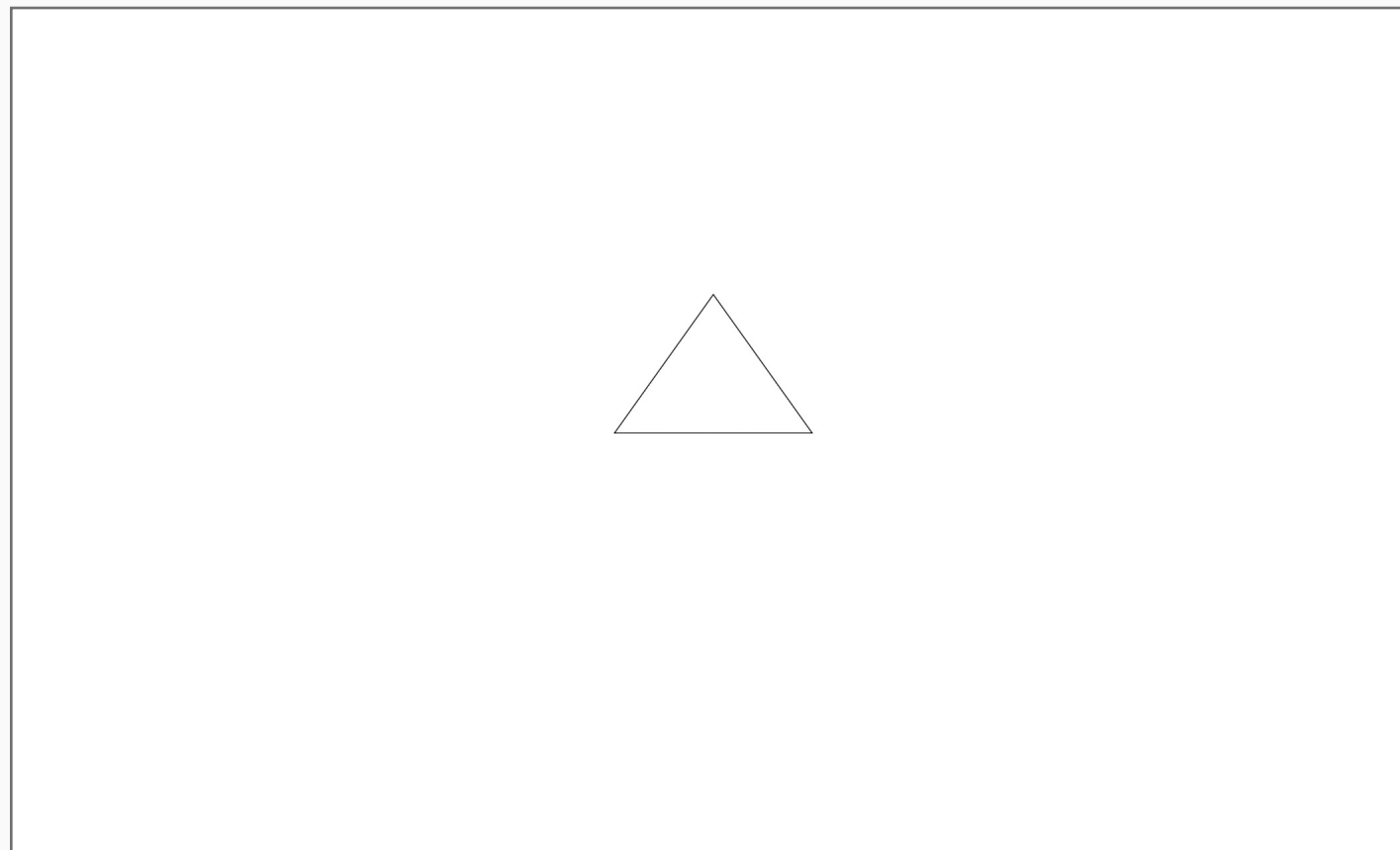
第四引数に縦幅を指定します（省略可）

※円と違って座標の指定が左端の頂点なので注意が必要です

```
rect(200, 300, 10, 10);
```

triangle()

Study

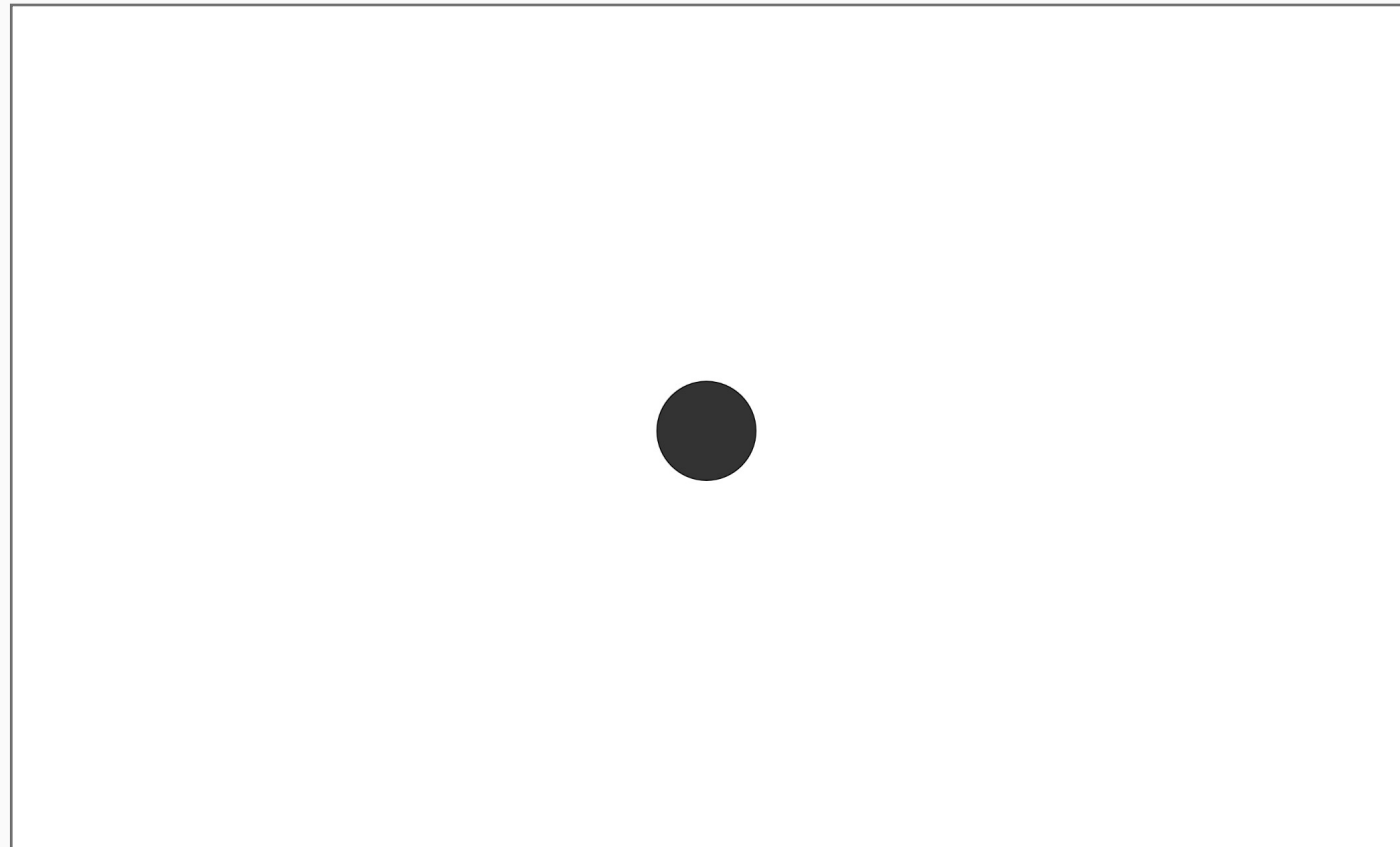


三角形の描画はtriangleメソッドによって行います。
引数は各頂点の座標です。

```
triangle(100, 100, 300, 100, 150, 50);
```

fill()

Study

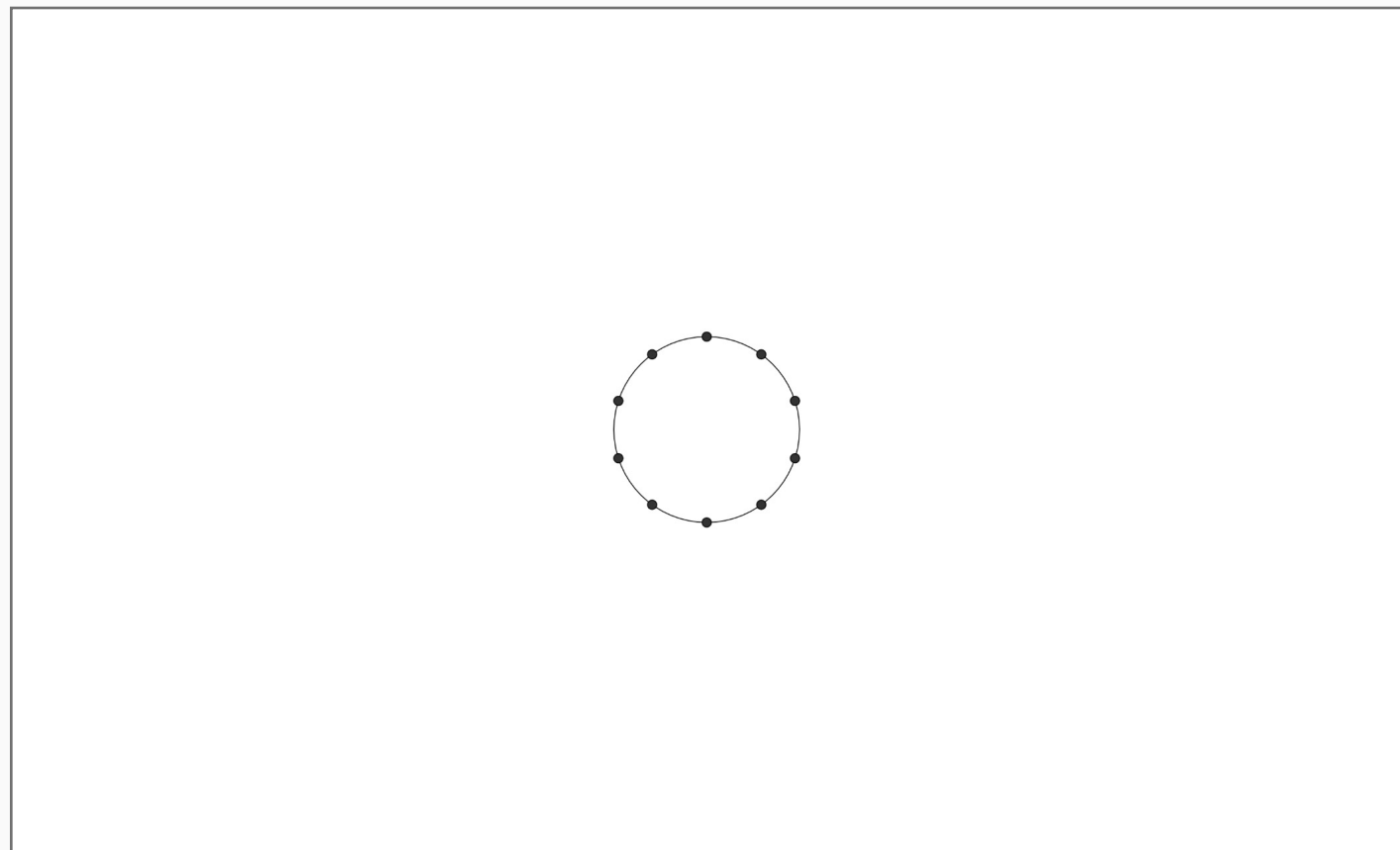


図形の塗り潰しはfillメソッドによって行います。

引数はredなどのキーワード、#222222などのカラーコードなど
要するにcssと同様の指定で使えます

```
fill("#333333");  
ellipse(200, 200, 100);
```

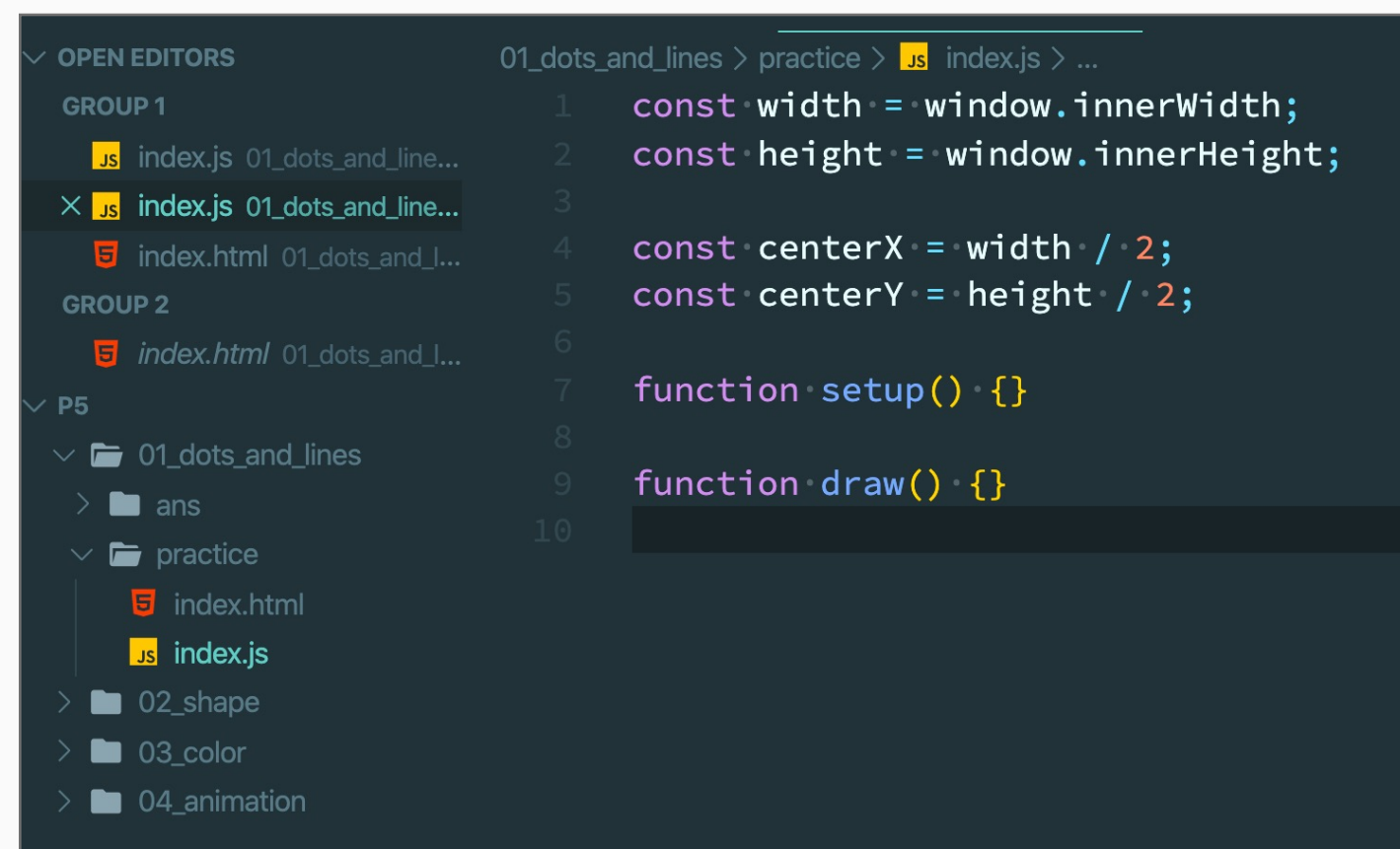
sin() cos()
radians()



円周上の座標はsin,cos, radiansメソッドによって求めます。
詳しい説明は割愛しますが、『弧度法』でググってみてください

```
const r = 100;  
const ellipseLength = 10;  
const angleStep = 360 / ellipseLength;  
  
ellipse(centerX, centerY, r * 2);  
for (let i = 0; i < ellipseLength; i++) {  
  const x = sin(radians(angleStep * i)) * r + centerX;  
  const y = cos(radians(angleStep * i)) * r + centerY;  
  ellipse(x, y, 10);  
}
```

Practice



```
01_dots_and_lines > practice > index.js > ...
1  const width = window.innerWidth;
2  const height = window.innerHeight;
3
4  const centerX = width / 2;
5  const centerY = height / 2;
6
7  function setup() {}
8
9  function draw() {}
10
```

ここまでの情報で図形の描画をしてみましょう

02_shape > practice > index.js

まとめ

shape

- ・ 図形は基本的に座標の指定で描画する
- ・ 円周上の座標を求める時はsin,cosを使う

色

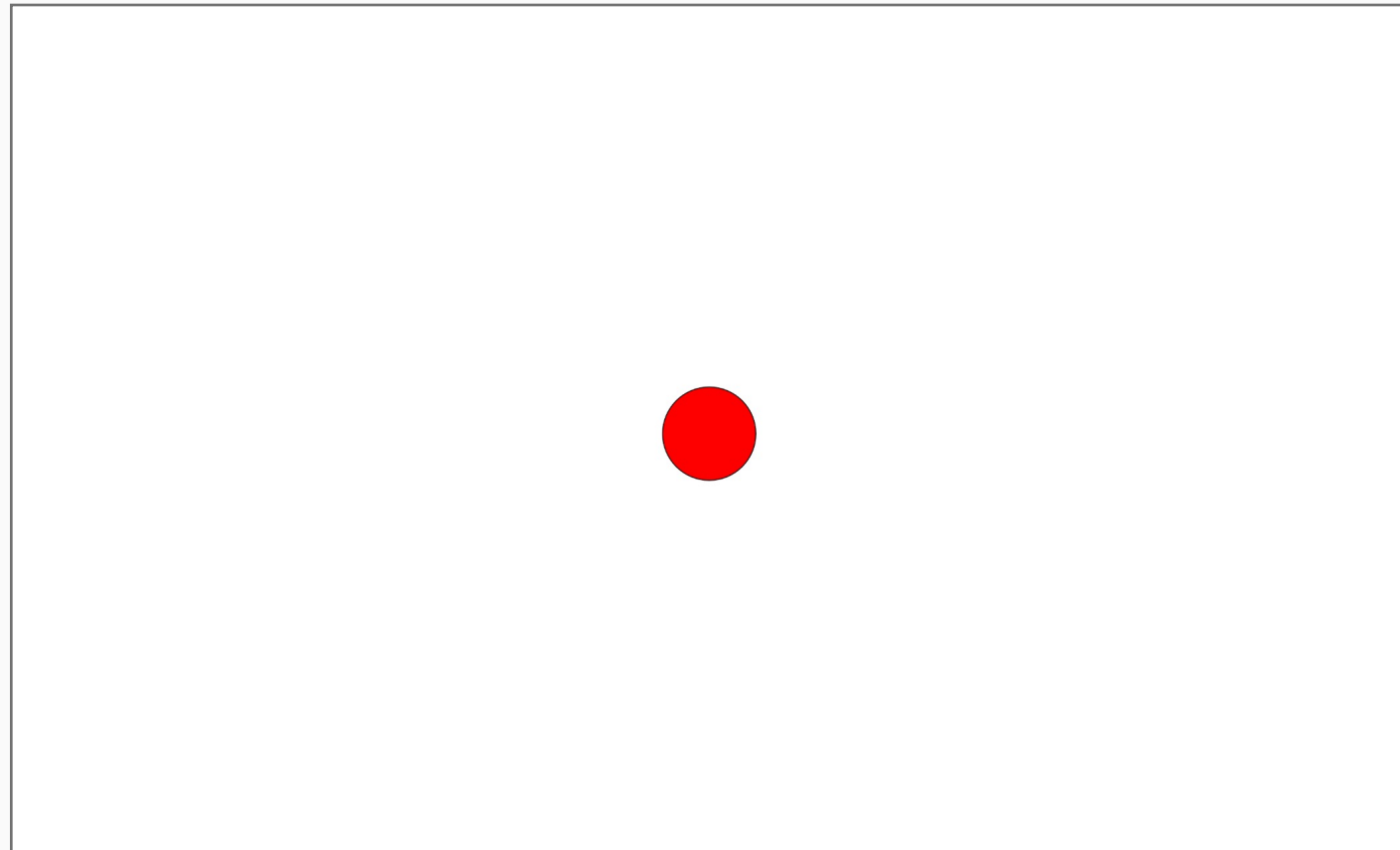
Color

色

Color

色の仕組みを理解する事は非常に重要です
このチャプターでは色についての概念と
p5.jsを通じて理論の実践を学びます

colorMode(HSB, 360, 100, 100)

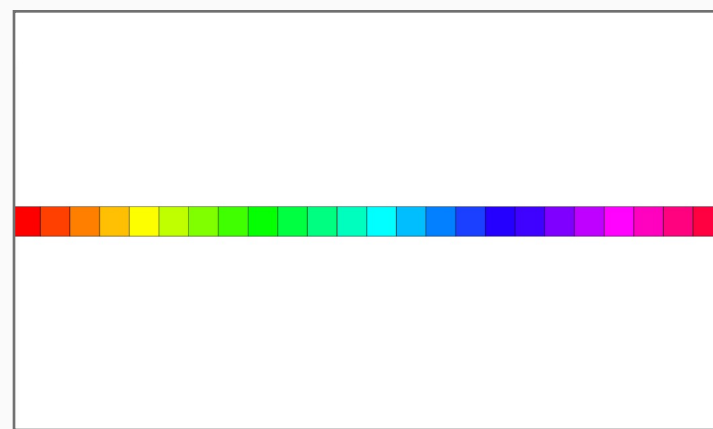


色の説明をするためにcolorModeメソッドで色の値の解釈方法を変更します。
引数は便宜的にHSB,360,100,100の四つを与えてください。
colorModeメソッドを実行した行以降は、fillなどの色のルールが全て変更されます

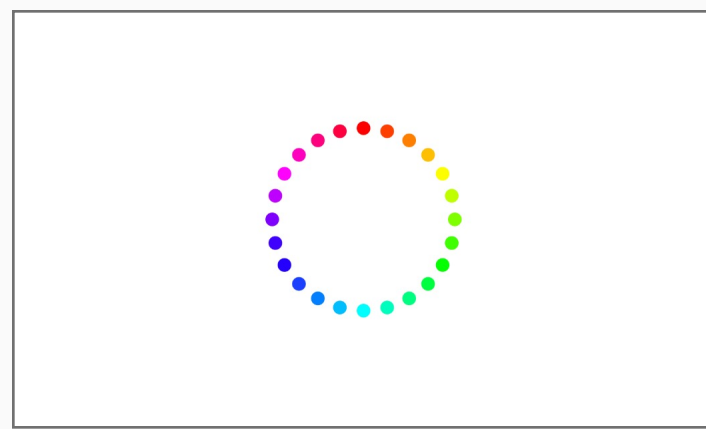
```
colorMode(HSB, 360, 100, 100)  
fill(360, 100, 100);  
ellipse(centerX, centerY, 100);
```

hue

Study



分光スペクトル



色相環

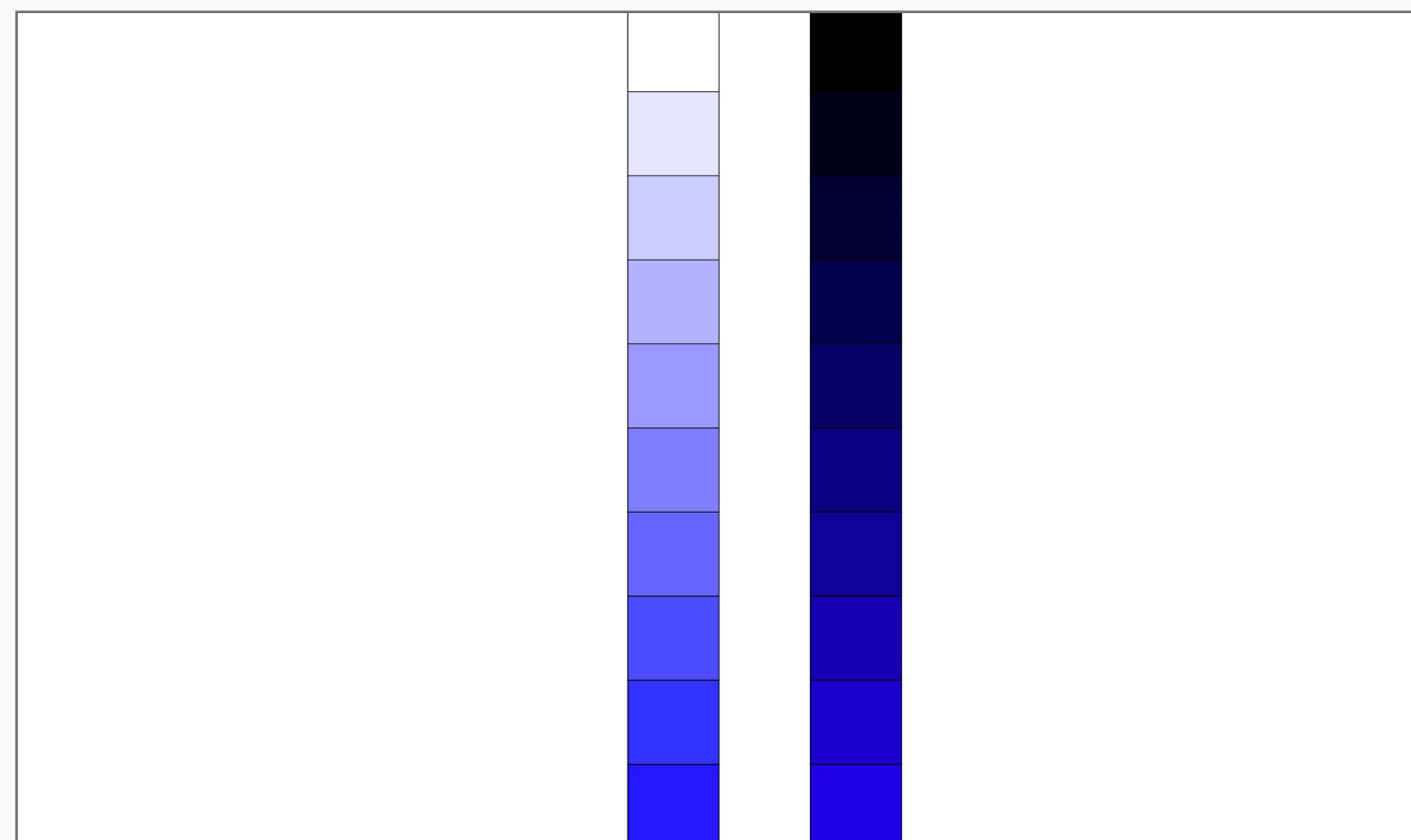
太陽光を波長毎に分けて作る分光スペクトルの最初（赤）と最後（紫）を繋げて円環状にしたものを『色相環』と呼びます。

RGBはそれぞれ 0° , 120° , 240° の位置にあり
色相環上で対角線にある色の関係が補色になります。

角度で色と関係のある色がわかりやすいのでHSBは配色で非常によく使われます

```
colorMode(HSB, 360, 100, 100)
```

Saturation Brightness



Saturation

Brightness

SaturationとBrightnessは原色の絵具に白と黒を混ぜたもの
と考えるとわかりやすい

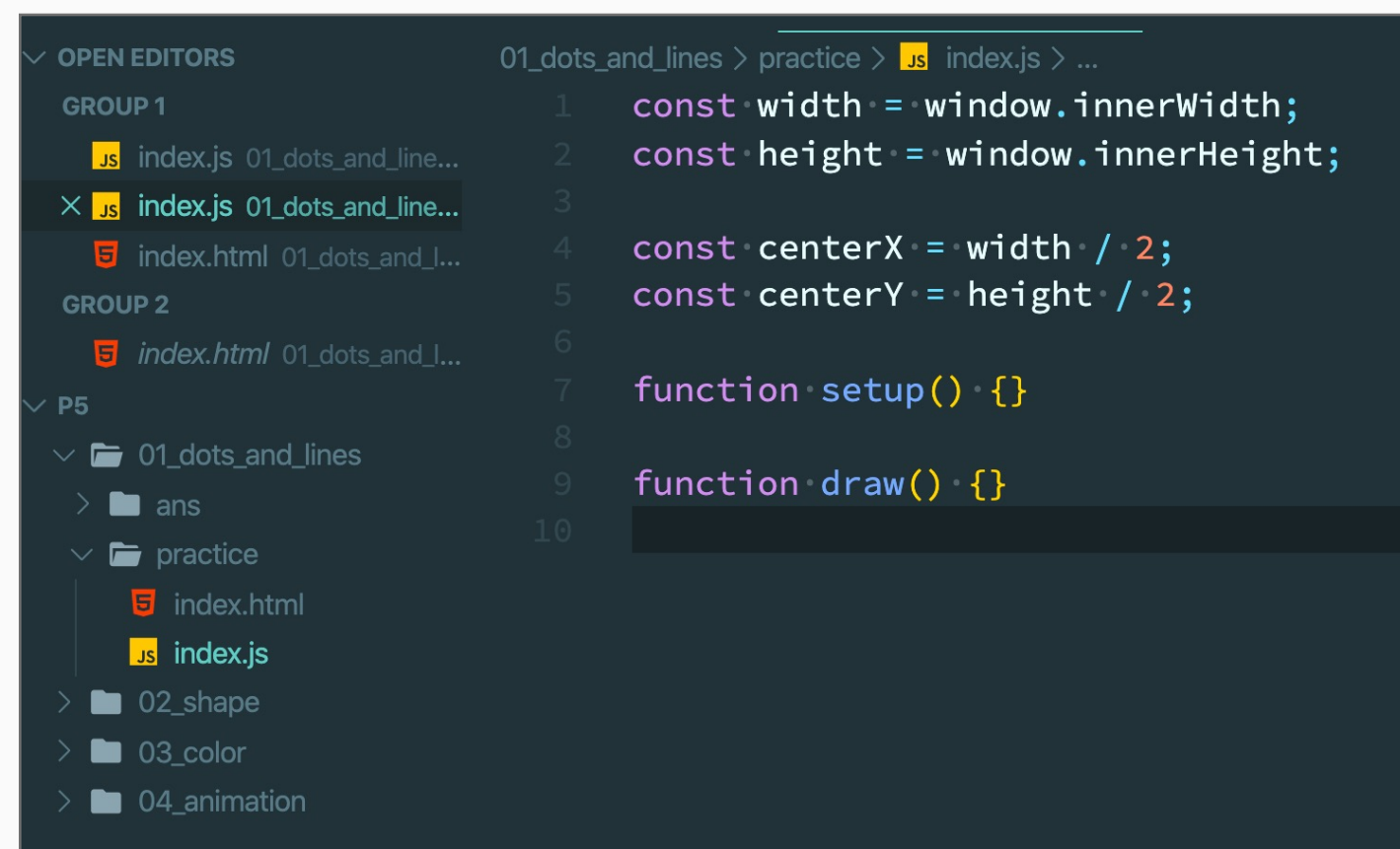
`colorMode(HSB, 360, 100, 100)`

の第二引数がsaturation

第3引数がbrightnessにそれぞれ対応している

`colorMode(HSB, 360, 100, 100)`

Practice



```
01_dots_and_lines > practice > index.js > ...  
1  const width = window.innerWidth;  
2  const height = window.innerHeight;  
3  
4  const centerX = width / 2;  
5  const centerY = height / 2;  
6  
7  function setup() {}  
8  
9  function draw() {}  
10
```

HSB形式の色の指定を試してみよう

03_color > practice > index.js

まとめ

Color

- ・色はHSB形式の表記にすると使う時に便利

動き

Animation

動き

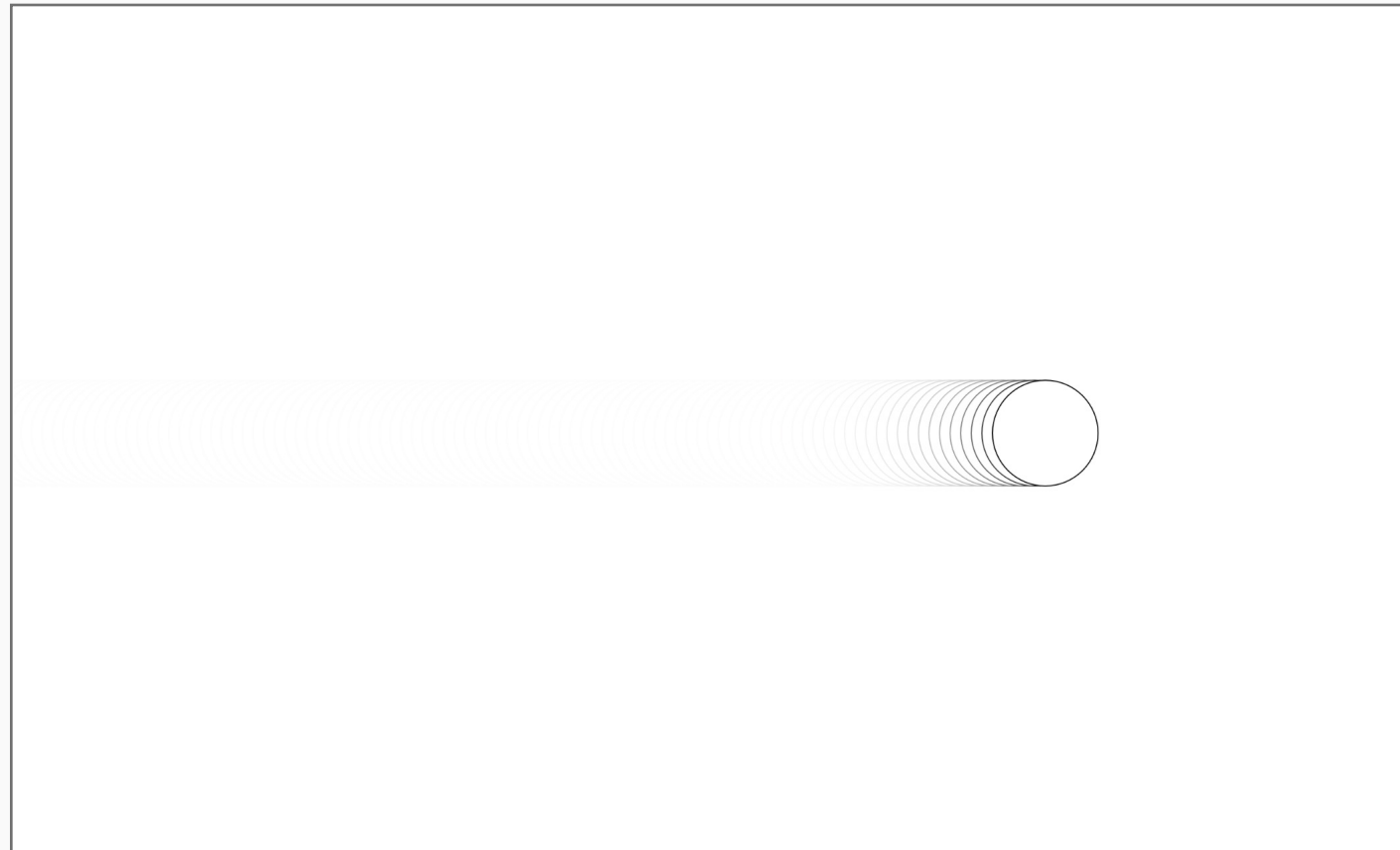
Animation

最後のチャプターは動きについてです。

アニメーションの仕組みとdraw関数について学びます

draw()

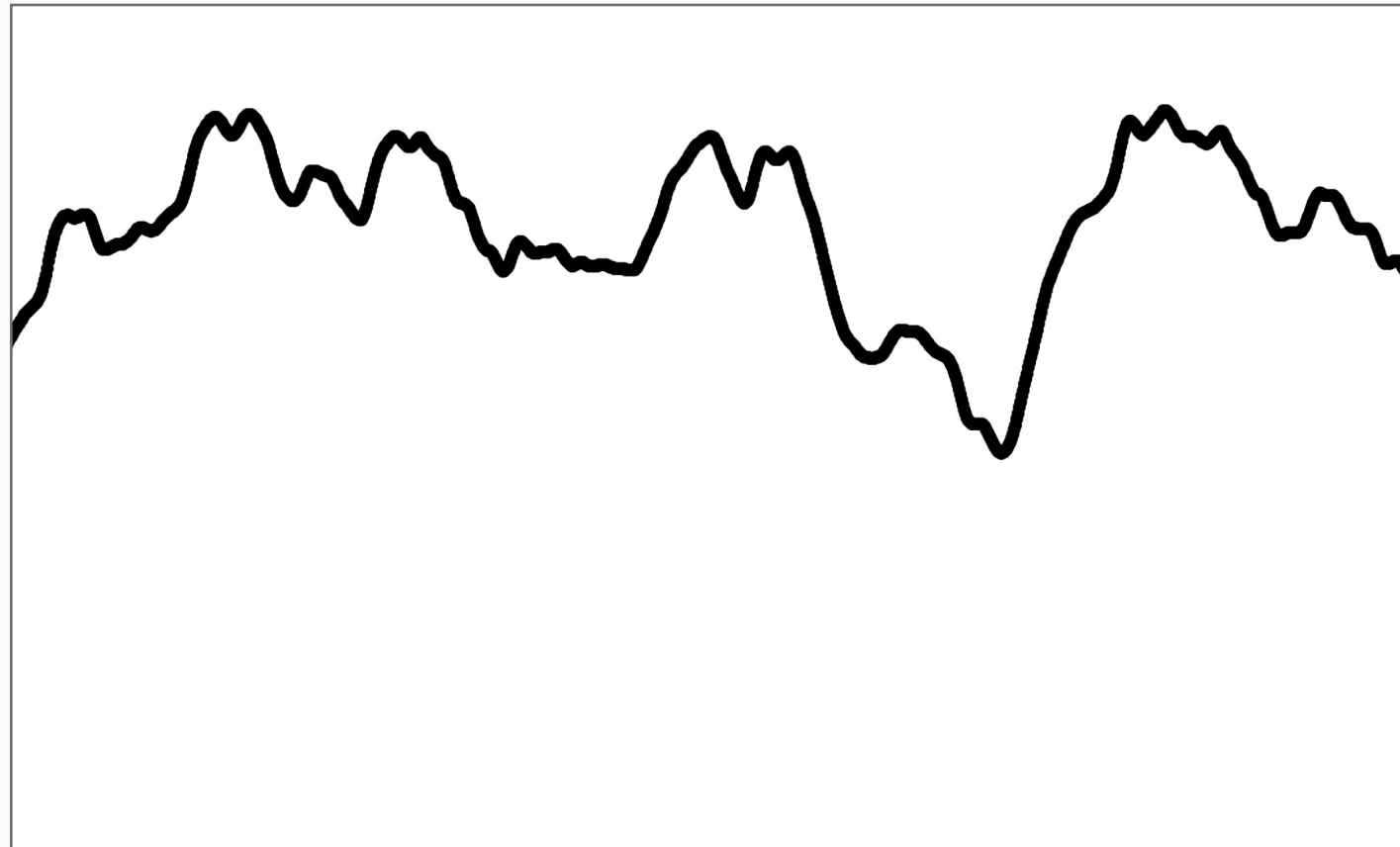
Study



要素をdraw関数内に記述すれば簡単にアニメーションします。
特別な指定がない限り、draw関数は1秒間に60回実行されます

```
let xStep = 0;  
function draw() {  
  xStep += 10;  
  ellipse(0 + xStep, centerY, 100);  
}
```

noise()



p5.jsには特別な乱数を生成するnoiseメソッドがあります。

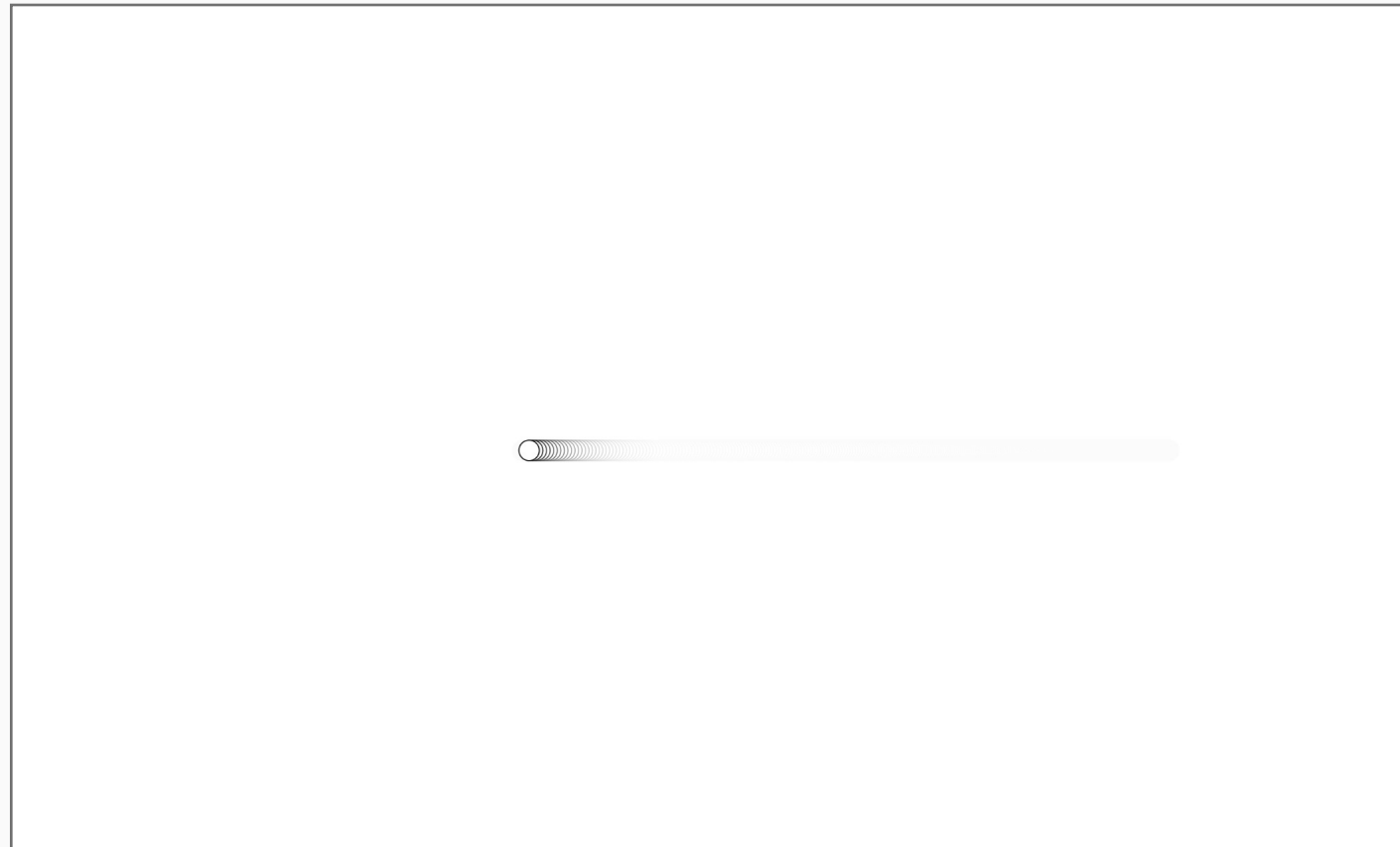
noiseメソッドはランダムでありながら、完全なランダムではない不思議なメソッドです

※詳しくは『パーリンノイズ』で検索してみてください

```
let xStep = 0;
let noiseStep = 0;
function draw() {
  xStep += 1;
  noiseStep += 0.01;
  const noiseVal = noise(noiseStep);
  console.log(noiseVal);
  point(xStep, noiseVal * centerY);
}
```

速度と加速度

Study



速度とは単位時間あたりに進む距離、

加速度は単位時間あたりに増える速度の量を指します。

単純な等速直線運動に飽きたら動きに加速度をつけてみてください。

```
let velocity = 1;
```

```
let acceleration = 0;
```

```
function draw() {
```

```
  velocity += acceleration;
```

```
  if (mouseIsPressed) {
```

```
    acceleration += 0.02;
```

```
  } else {
```

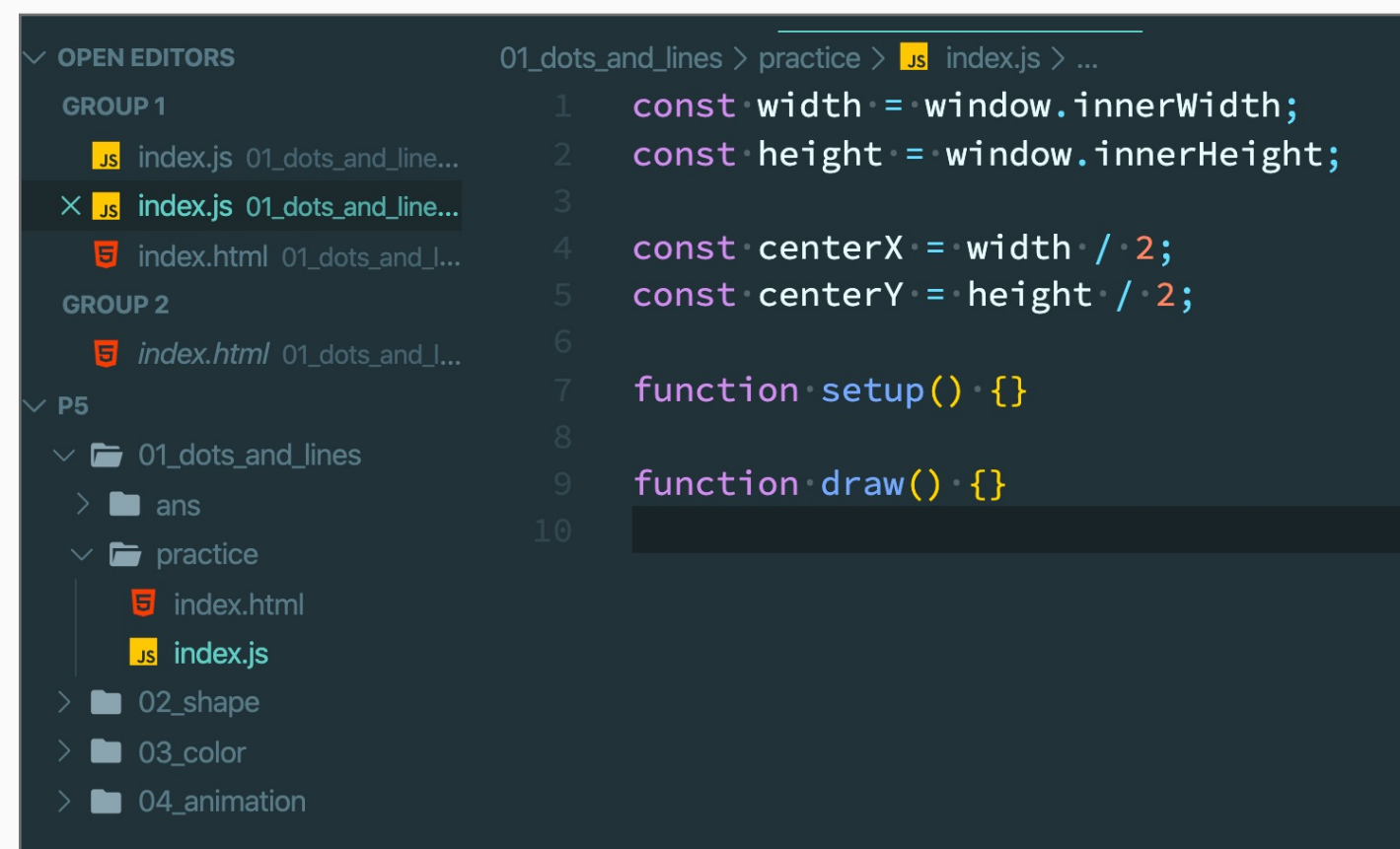
```
    acceleration -= 0.01;
```

```
  }
```

```
  ellipse(centerX + velocity, centerY, 20);
```

```
}
```

Practice



```
01_dots_and_lines > practice > .js index.js > ...
1  const width = window.innerWidth;
2  const height = window.innerHeight;
3
4  const centerX = width / 2;
5  const centerY = height / 2;
6
7  function setup() {}
8
9  function draw() {}
10
```

Draw関数を使ってアニメーションさせてみよう

04_animation > practice > index.js

まとめ

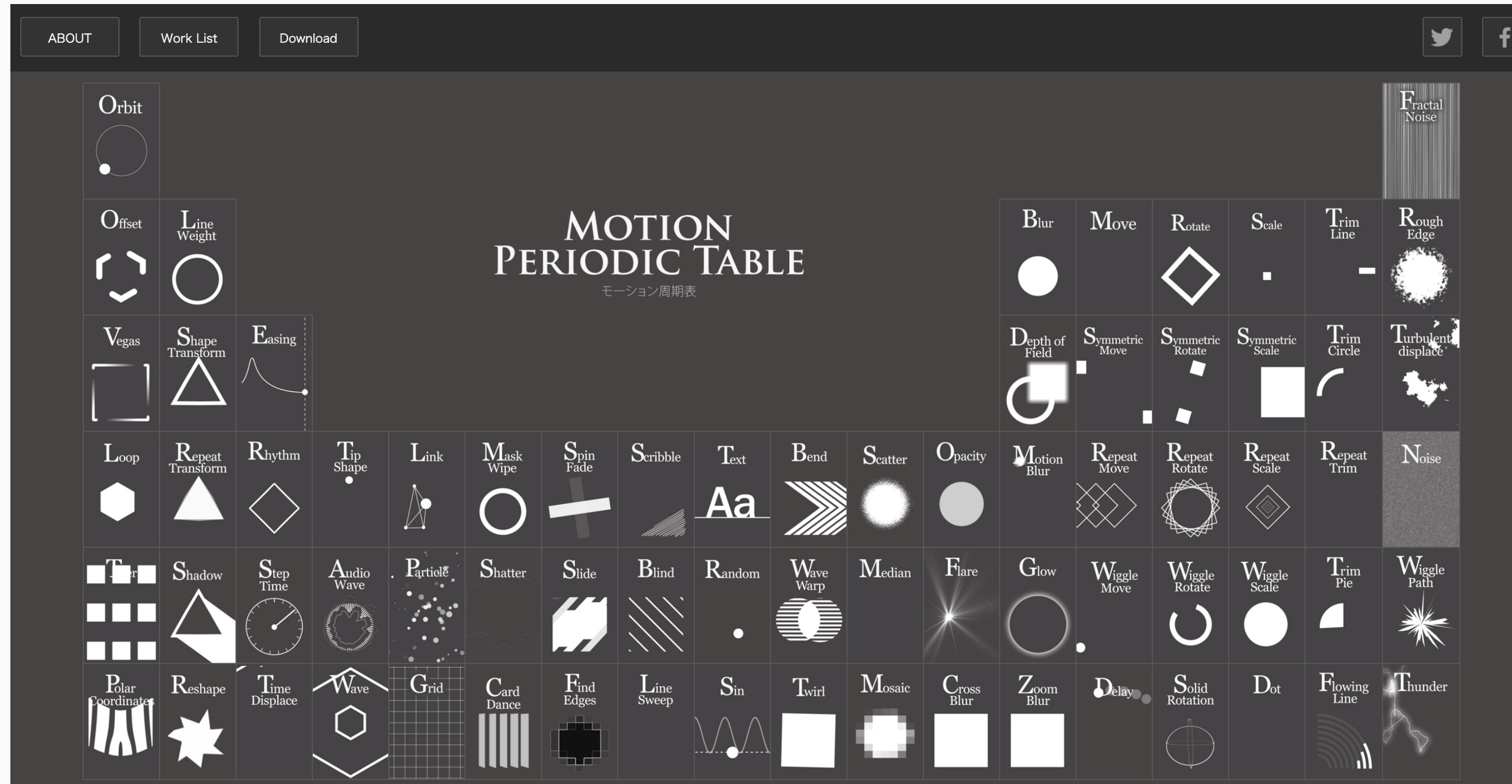
Animation

- draw関数は一秒間に60回実行される
- draw関数の外で定義した変数をインクリメントする
- その変数を何かに当てはめるとアニメーションする

演習

Practice

Generative Art - practice -



<http://foxcodex.html.xdomain.jp/>

各モーションを観察して実装する



- 動きの種類（移動・回転・拡大縮小）
- 時間に対する動きの量
- 動きの組み合わせ
- これらに注目して実装してみる

参考資料

P5.js リファレンス: <https://p5js.org/reference/>

yoppa.org: <https://yoppa.org/>

普及版ジェネラティブ・アート: www.amazon.co.jp/dp/4861009634

Nature of code : www.amazon.co.jp/dp/4862462456

数学から創るジェネラティブアート : www.amazon.co.jp/dp/B07QLC3PX5

Generative Design with p5.js : www.amazon.co.jp/dp/B07DQHV9GX

ご清聴ありがとうございました