p5.js を使った javascript体験

まず editor p5js を開いておこう

Google 検索

キーワード

editor web p5

次に Github を開いておこう

Github

github.com/amami-harhid

STEP 1：キャンバスを作り出す

STEP 1：キャンバスを作る

createCanvas(幅, 高さ)

```
function setup() {
    createCanvas(400,400);  // ⇒  100, 100 にしてみよう
}
function draw() {
    background(220);
}
```

createCanvasのパラメータの数字を、いろいろな数字に変えてみよう

STEP 2：変数で幅と高さを指定する

変数、それはプログラミングの大切な考え方です

STEP02: 変数で幅と高さを指定する

```
const W = innerWidth; // コンテンツを表示する領域の横の長さ
const H = innerHeight; // コンテンツを表示する領域の縦の長さ
function setup() {
    createCanvas(W, H);
}
function draw() {
    background(220);
}
```

STEP02: 変数で幅と高さを指定する

```
const W = innerWidth; // コンテンツを表示する領域の横の長さ
const H = innerHeight; // コンテンツを表示する領域の縦の長さ
function setup() {
    createCanvas(W, H);
}
function draw() {
    background(220);
}
```

STEP02: 変数で幅と高さを指定する

```
const W = innerWidth; // コンテンツを表示する領域の横の長さ
const H = innerHeight; // コンテンツを表示する領域の縦の長さ
function setup() {
    createCanvas(W, H);
}
function draw() {
    background(220);
}
```

STEP02: 変数で幅と高さを指定する

```
const W = innerWidth; // コンテンツを表示する領域の横の長さ
const H = innerHeight; // コンテンツを表示する領域の縦の長さ
function setup() {
    createCanvas(W, H);
}
function draw() {
    background(220);
}
```

STEP 2：変数で幅と高さを指定する

const

固定値を定義するときのキーワードです。

const で定義した変数は、一度格納した値は変更することができなくなります。

```
const W = innerWidth;
```

STEP 2：変数で幅と高さを指定する

```
const W = innerWidth;
```

と定義すると、innerWidthの変数に入っている値を、W へ入れるという意味になります。

```
const 固定値の名前 = 固定値のデータ
```

STEP 2：変数で幅と高さを指定する

window.**innerWidth** と　window.**innerHeigth**

inner　は 内側という意味
width　は 横幅という意味
Height は 高さ という意味

STEP 2：変数で幅と高さを指定する

window.**innerWidth** と　window.**innerHeigth**

inner　は 内側という意味
width　は 横幅という意味
Height は 高さ という意味

STEP 2：変数で幅と高さを指定する

window.**innerWidth** と　window.**innerHeigth**

inner　は 内側という意味
width　は 横幅という意味
Height は 高さ という意味

STEP 2：変数で幅と高さを指定する

window.**innerWidth** と　window.**innerHeigth**

inner　は 内側 という意味
width　は 横幅という意味
Height は 高さ という意味

STEP 2：変数で幅と高さを指定する

window.**innerWidth** と　window.**innerHeigth**

inner　は 内側という意味
width　は 横幅という意味
Height は 高さ という意味

STEP 2：変数で幅と高さを指定する

window.**innerWidth** と　window.**innerHeigth**

inner　は 内側という意味
width　は 横幅という意味
Height は 高さ という意味

コンテンツを表示する領域の幅と高さのことです。

Javascript では、window オブジェクトで innerWidth, innerHeightを参照できます。

参照するときは、ドット(.) でつなげて書きます。

例： window.innerWidth

Javascript では、window オブジェクトで innerWidth, innerHeightを参照できます。

参照するときは、ドット(.) でつなげて書きます。

例： window.innerWidth

windowオブジェクト

画面上に表示されているすべてのオブジェクトをとりまとめるオブジェクトのことです。

javascriptでは、 window を省略できますので、
window.**innerWidth**, window.**innerHeight** は、

innerWidth, **innerHeight** と書くことができます。

windowオブジェクト

画面上に表示されているすべてのオブジェクトをとりまとめるオブジェクトのことです。

javascriptでは、 window を省略できますので、window.**innerWidth**, window.**innerHeight** は、

innerWidth, **innerHeight** と書くことができます。

STEP 3：**背景色を変える**

STEP 3：背景色を変える

```
function draw() {
    background(220,220,220); // ⇒ 220 のところを変えてみよう
}
```

background(赤, 緑, 青)

赤, 緑, 青 は 0 ～ 255 の範囲

STEP 3：背景色を変える

```
function draw() {
    background(220,220,220); // ⇒ 220 のところを変えてみよう
}
```

background(赤, 緑, 青)

赤, 緑, 青 は 0 ～ 255 の範囲

STEP04：変数を使って色を指定

STEP04：変数を使って色を指定

```
let r , g, b;
```

```
function setup() {
    createCanvas(W, H);
    r = 220; g = 220; b = 220;
}
```

変数 r , g, b を定義し、setup の中で値をいれています

STEP04：変数を使って色を指定

```
function draw() {
    background(r, g, b);
}
```

background(赤, 緑, 青)

変数 r , g, b は setup の中で値をいれていますので
background の 赤, 緑, 青 のパラメータとして
使うことができるというわけです

STEP04：変数を使って色を指定

```
let r , g, b;
```

let の解説をします

let　変数の名前 で変数を定義できます

const と違い、データ内容を変更できます。

変数の定義は『 , 』でつなげてもOK

STEP04：変数を使って色を指定

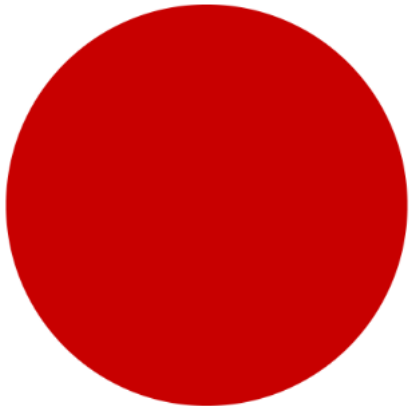
```
let r = 0;
```

let　変数の名前 = 値;

と書くと 変数を定義して、同時に
値も入れてしまう、という意味です。

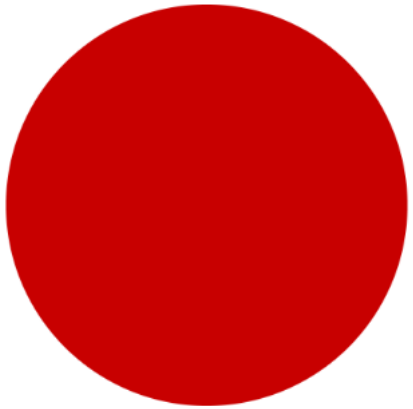
STEP05：円を描く

STEP05：円を描く



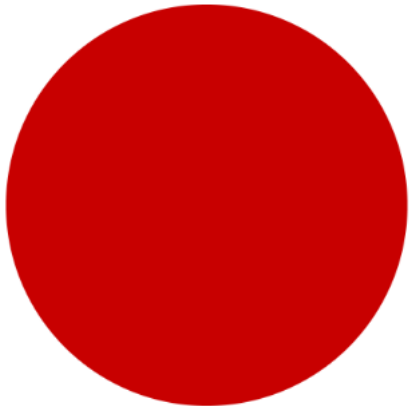
円の中心座標、円の半径 を使って 円を指定する

STEP05：円を描く



円の中心座標、円の半径 を使って 円を指定する

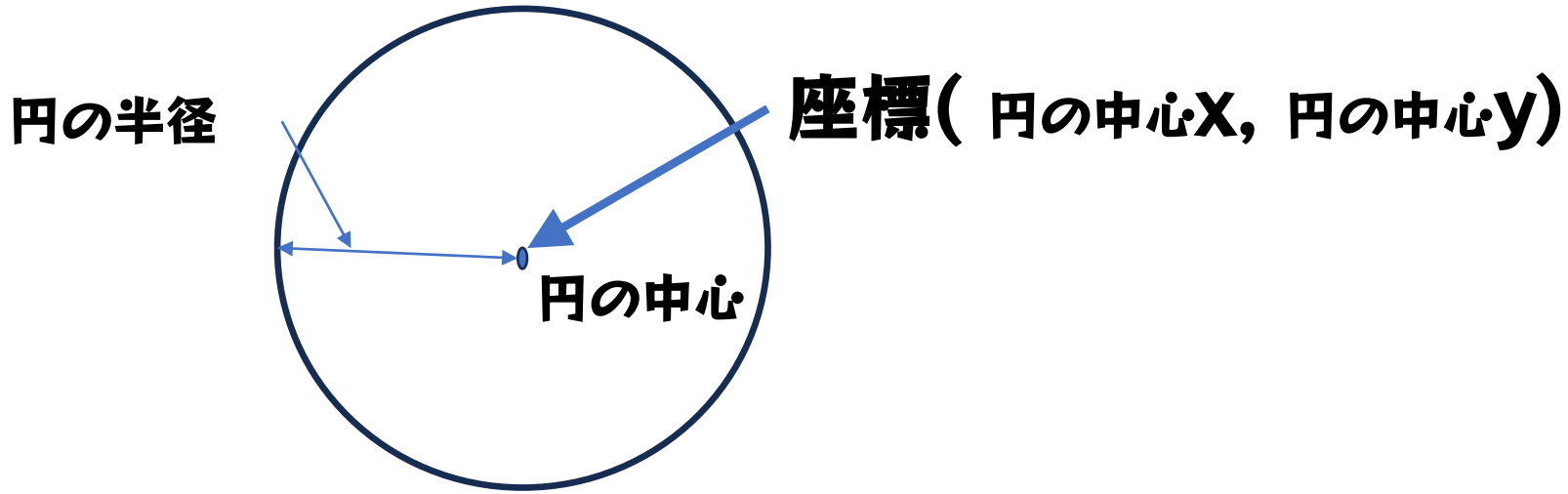
STEP05：円を描く



円の中心座標、円の半径 を使って 円を指定する

STEP05：円を描く

ellipse(円の中心x, 円の中心y, 円の半径);



STEP05：円を描く

```
const W = innerWidth;
const H = innerHeight;
let r , g, b;
function setup() {
    createCanvas(W, H);
    r = 220; g = 220; b = 220;
}
function draw() {
    background(r, g, b);
    ellipse( 200, 200, 50 );
}
```

STEP06：円を描く(変数)

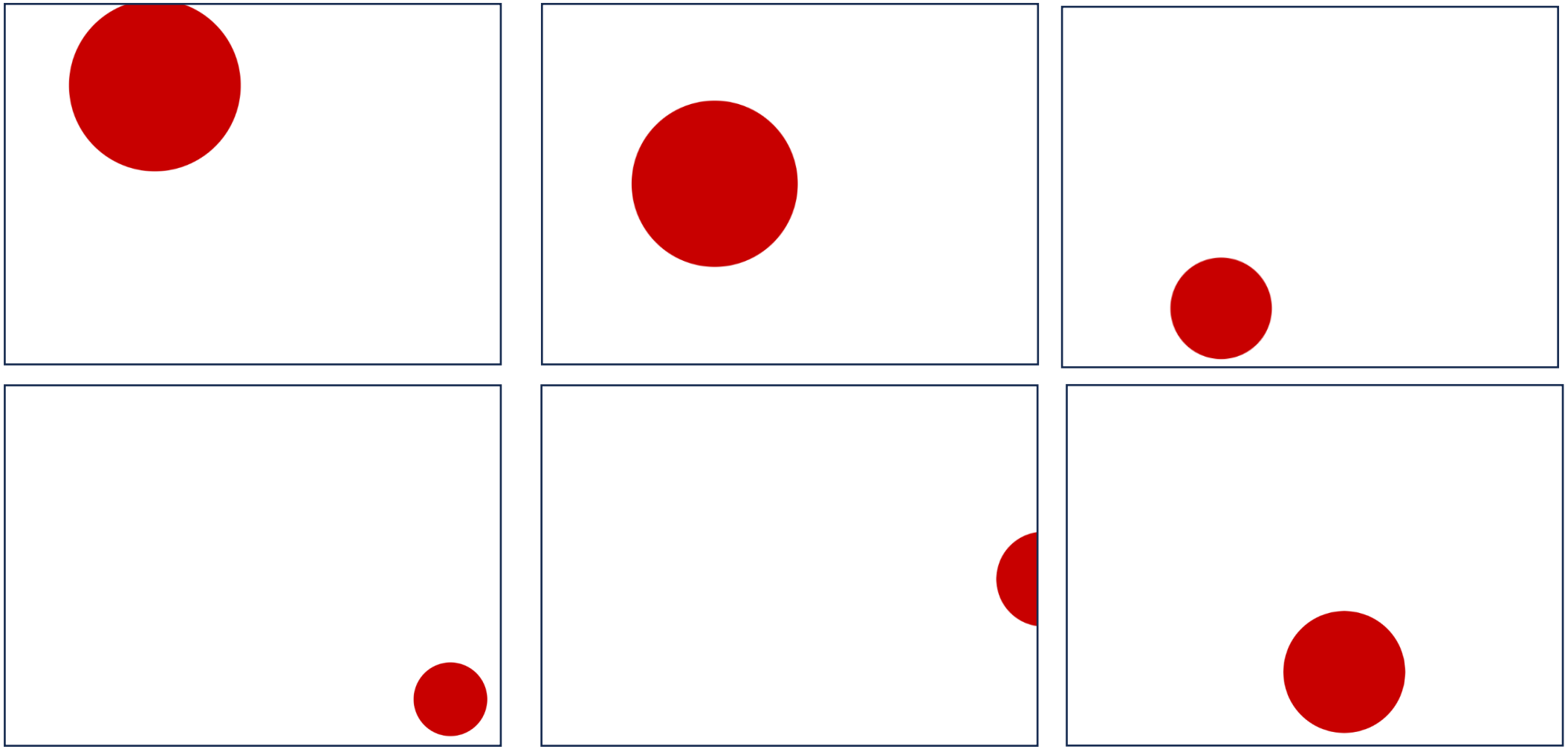
円の中心座標、円の半径 を変数にする

STEP06：円を描く(変数)

```
function draw() {
    background(r, g, b);
    // ↓ 一時的に使う変数なので _ をつけた。
    let _x = 100;
    let _y = 200;
    let _r = 100;
    ellipse( _x, _y, _r );
}
```

STEP07：**ランダム関数**

円の中心、円の半径をランダムに指定してみる



円の中心、円の半径が変化する

random(最小値, 限界値)

最小値～限界値の範囲でランダム(限界値含まず)

random(限界値)

0～限界値の範囲でランダム (限界値含まず)

random( )

0～1 の範囲でランダム (1は含まず)

random(最小値, 限界値)

最小値～限界値の範囲でランダム(限界値含まず)

random(限界値)

0～限界値の範囲でランダム (限界値含まず)

random( )

0～1 の範囲でランダム (1は含まず)

frameRate(数字)

FPS を指定 （ 0 より大きな数 ）

FPSを小さくする用途として使う

P5.js では FPS=30 より大きいときは
その効果を感じることはできない。
FPS=30が最大みたいです。

frameRate(数字)

FPS を指定 （ 0 より大きな数 ）

FPSを小さくする用途として使う

P5.js では FPS=30 より大きいときは
その効果を感じることはできない。
FPS=30が最大みたいです。

STEP07：ランダム関数

```
function draw() {
  background(r,g,b);
  // ↓ ここに円を描くコードを記述する
  let _x = random(0,W);       // 0 ～ W の範囲でランダムに値を決める
  let _y = random(0,H);       // 0 ～ H の範囲でランダムに値を決める
  let _r = random(10,100);    // 10 ～ 100 の範囲でランダムに値を決める
  ellipse(_x,_y,_r);
}
```

STEP08：円の色を変える

color 関数で 色を作る
fill 関数で塗り潰す色を指定
stroke,noStroke 関数で
枠線の色を指定

STEP08：円の色を変える

color 関数で 色を作る
fill 関数で塗り潰す色を指定
stroke,noStroke 関数で
枠線の色を指定

STEP08：円の色を変える

color 関数で 色を作る
fill 関数で塗り潰す色を指定
stroke,noStroke 関数で
枠線の色を指定

STEP08：円の色を変える

color 関数で 色を作る
fill 関数で塗り潰す色を指定
stroke,noStroke 関数で
枠線の色を指定

color(数)

灰色になります (0 – 255)

color(赤, 緑, 青)

色の重ね合わせです (0 – 255)

color(赤, 緑, 青, アルファ)

透明度効果を与えます(0 – 255)

color(数)

灰色になります (0 – 255)

color(赤, 緑, 青)

色の重ね合わせです (0 – 255)

color(赤, 緑, 青, アルファ)

透明度効果を与えます(0 – 255)

STEP08：**color 関数**

```
const c = color(
  random(150,255),
  random(150,255),
  random(150,255)
);
```

fill(色)

この後に描画される図形の
塗り潰しの色を指定する

stroke(色)

この後に描画される図形の
枠線の色を指定する

noStroke ( )

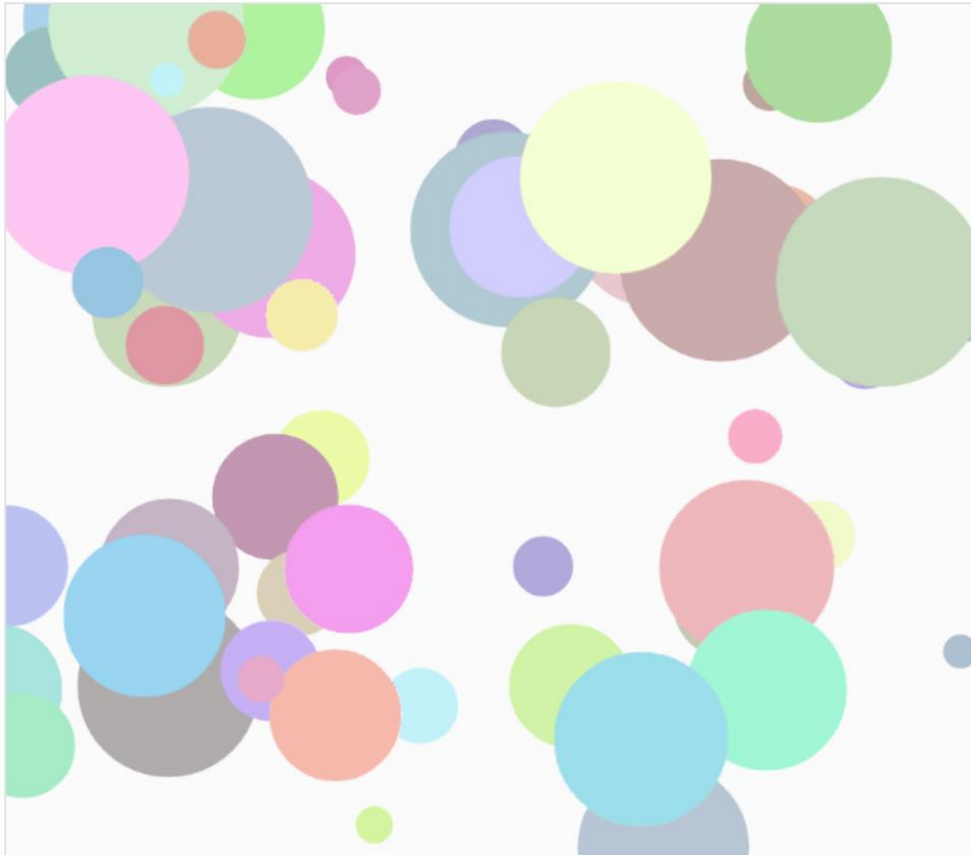
この後に描画される図形の
枠線は「なし」とする

STEP08：
円の色を変える

```
function draw() {
  background(r,g,b);
  // ↓ ここに円を描くコードを記述する
  let _x = random(0,W)
  let _y = random(0,H)
  let _r = random(10,100)
  const c = color(
    random(150,255),
    random(150,255),
    random(150,255)
  );
  fill(c);
  noStroke();
  ellipse(_x,_y,_r);
}
```

STEP09：残像を残そう

STEP09：残像を残そう



STEP09：残像を残そう

background(r, g, b, ○)

色の指定の後に 数字をつけると
残像が残る時間が決まる。

0 のときは 残像が残るが 消えない

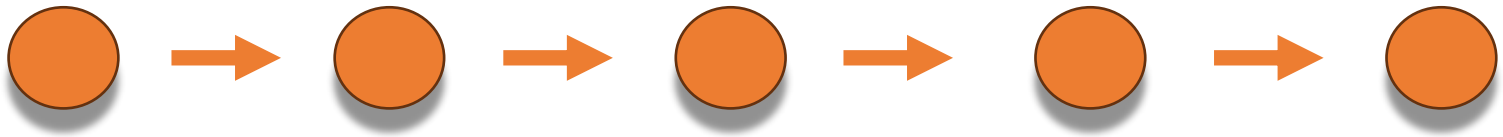
```
background(r, g, b, 0 );
```

50 のときは 少しの間だけ残像が残る

```
background(r, g, b, 50 );
```

STEP10：**円の位置を動かす**

STEP10：**円の位置を動かす**



STEP10：円の位置を動かす

```
function draw() {
    ;
    (途中省略)
    ;
    ellipse(_x,_y,_r);
}
```

変数 _x, _y, _r の値を変えることで
位置や円の大きさが変わります。

STEP10：円の位置を動かす

右側へ連続して動かす

変数 _x の値を増やしていくと
右側へ移動する

```
x = x + 5;
```

xの値に5を足して x に入れなおす

```
x += 5;
```

上と同じ意味の書き方

STEP10：円の位置を動かす

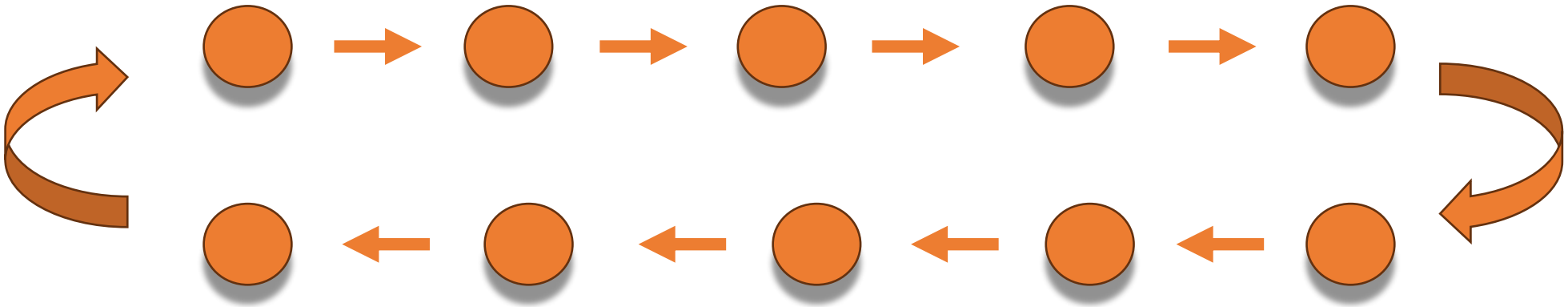
右側へ連続して動かす

変数 _x の値を増やしていくと右側へ移動する

```
let _x = W/2;
let _y = H/2;
let _r = 50;
function draw() {
   ;
   (途中省略)
   ;
   ellipse(_x,_y,_r);
   _x += 5; // _x を 5ずつ増やす、_x = _x + 5 と同じ意味です
}
```

STEP11：端についたら反対方向へ

STEP11：端についたら反対方向へ



STEP11：端についたら反対方向へ

右側へ連続して動かす

円の中心が端についたら
反対方向へ動く

動く方向を変数で指示

```
let  _direction = 1;

_x  += 5  *  _direction;

if ( _x > W  ||  _x < 0 ) {
   _direction  *=  -1;
}
```

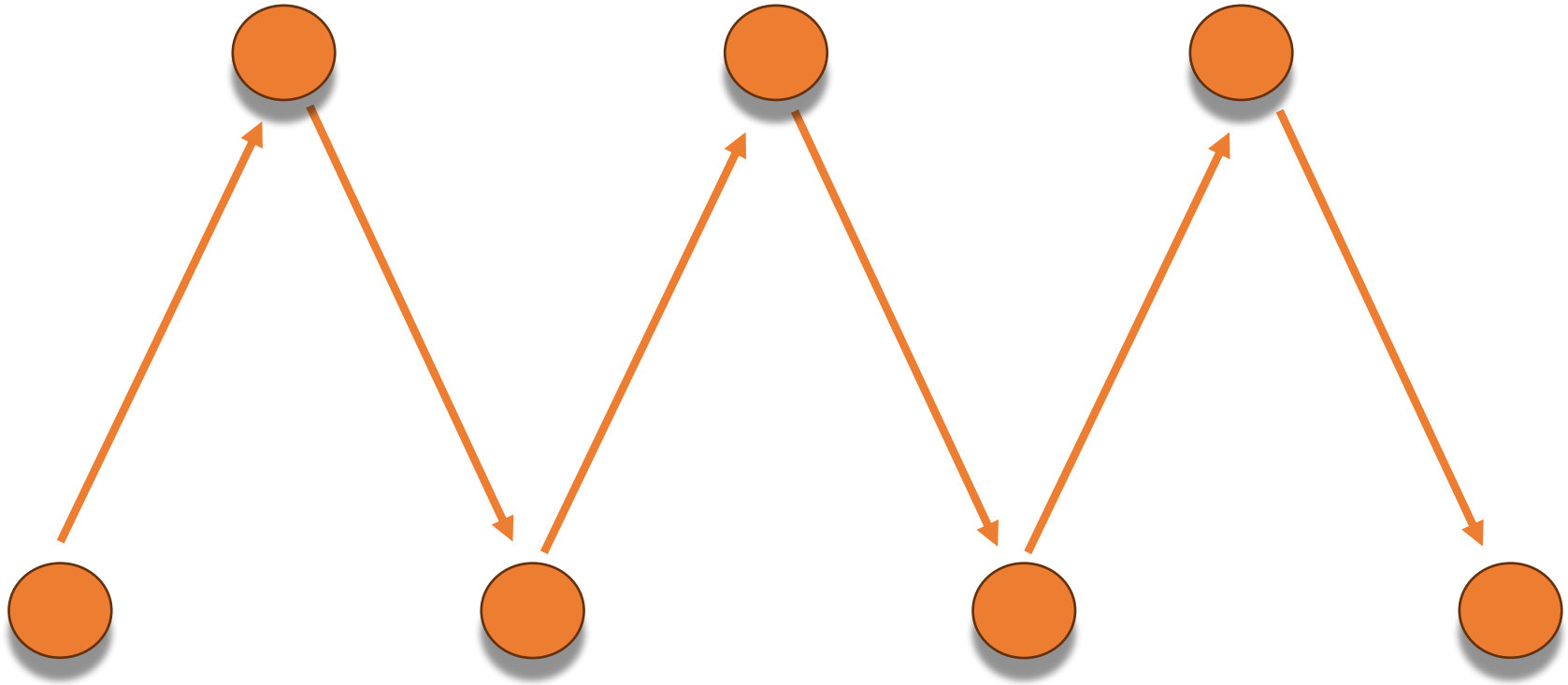
STEP11：端についたら反対方向へ

円の中心が端についたら反対方向へ動く

```
let _x = W/2;
let _y = H/2;
let _r = 50;
let _direction = 1;
function draw() {
   ;
   (途中省略)
   ;
    ellipse(_x,_y,_r);
    _x += _direction * 5; // * は掛け算です。
    // もしも _x が 0 より小さく、または、W より大きいとき
    if( _x < 0 || W < _x ) {
        // プラスのときはマイナスに、マイナスのときはプラスにする。
        _direction *= -1; // _direction = _direction * (-1) と同じ
    }
}
```

STEP12：上下に等速でジャンプ

STEP12：上下に等速でジャンプ



STEP12：上下に等速でジャンプ

```
let  _directionX = 1;
let  _directionY = 1;

_y += _directionY * 20;

if ( _y < H / 2  ||  H < _y ) {
  _directionY  *=  -1;
}
```

方向を変えるための変数を X方向, Y方向 用に作る

X 方向のコードと同じ感じで Y方向のコードをつくる。

H < _y ⇒ 下の端を超えた の意味

_y < H / 2 ⇒ 半分の高さ を超えた の意味

/ は割り算の意味

STEP12：上下に等速でジャンプ

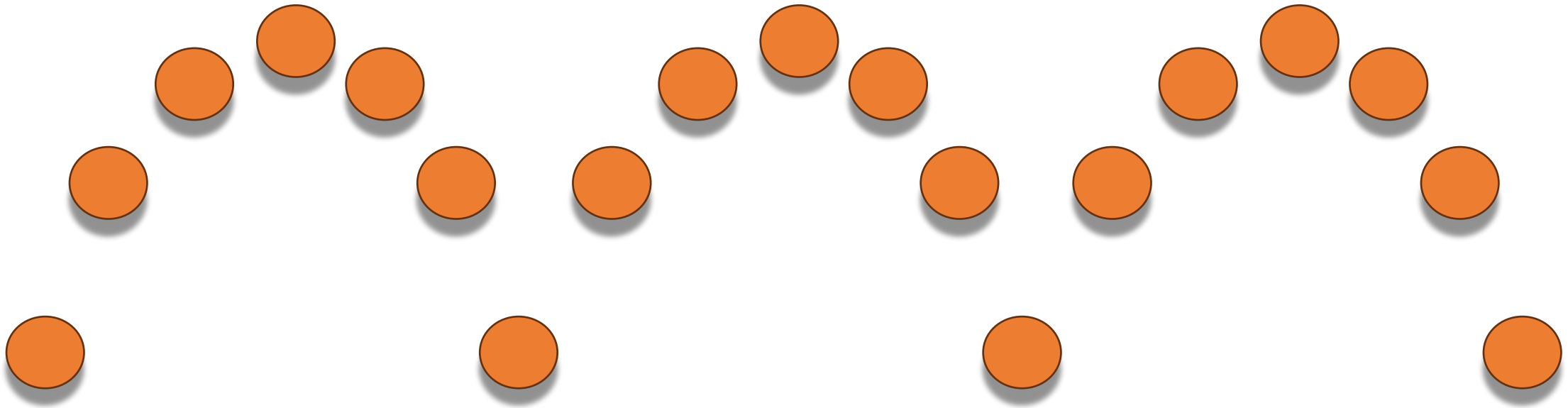
```
let _x = 0;
let _y = H/2;
let _r = 50;
let _directionX = 1;
let _directionY = 1;
```

```
function draw() {
  background(r,g,b);
  // ↓ ここに円を描くコードを記述する
  fill('red');
  stroke('red');
  ellipse(_x,_y,_r);
  _x += _directionX * 5;
  _y += _directionY * 20;
  if(_x < 0 || W < _x) {
    _directionX *= -1;
  }
  if(_y < H/2 || H < _y) {
    _directionY *= -1;
  }
}
```

色は 英語でも指定できる
red, green, blue, など

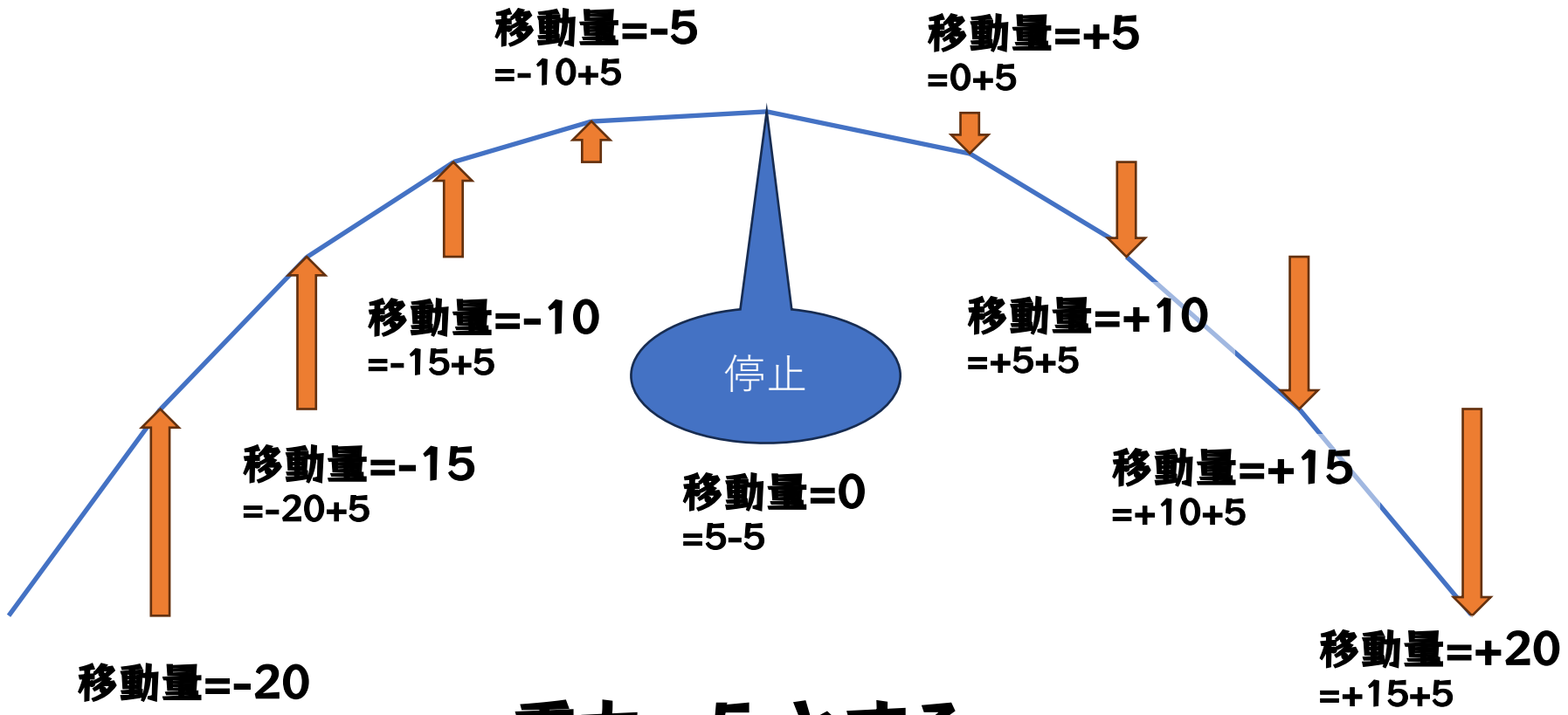
STEP13：**自由落下で跳ねる**

STEP13：**自由落下で跳ねる**



自由落下

速度(移動量)が (一定値で)変化していく



重力= 5 とする

自由落下

速度(移動量)が (一定値で)変化していく

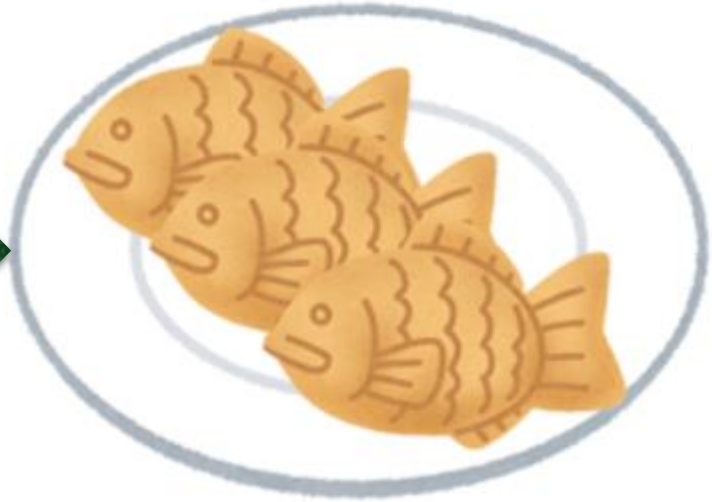
```
let _speed = _power; // 速度(移動量)
let _power = -25; // 初期速度( 上向き )
function draw() {
  background(r,g,b);
  // ↓ ここに円を描くコードを記述する
  fill('red');
  stroke('red');
  ellipse(_x,_y,_r);
  _x += _directionX * 5;
  if(_x < 0 || W < _x) {
    _directionX *= -1;
  }
  _y += _speed; // _y が小さいほど 上に行く
  _speed -= -1;  // 重力を -1 としている。
  if(_y > H) {
    _speed = _power;
  }
}
```

STEP14：**インスタンス化**

インスタンスは
クラスをひな形にして作られる

FUNCTION

new



クラス

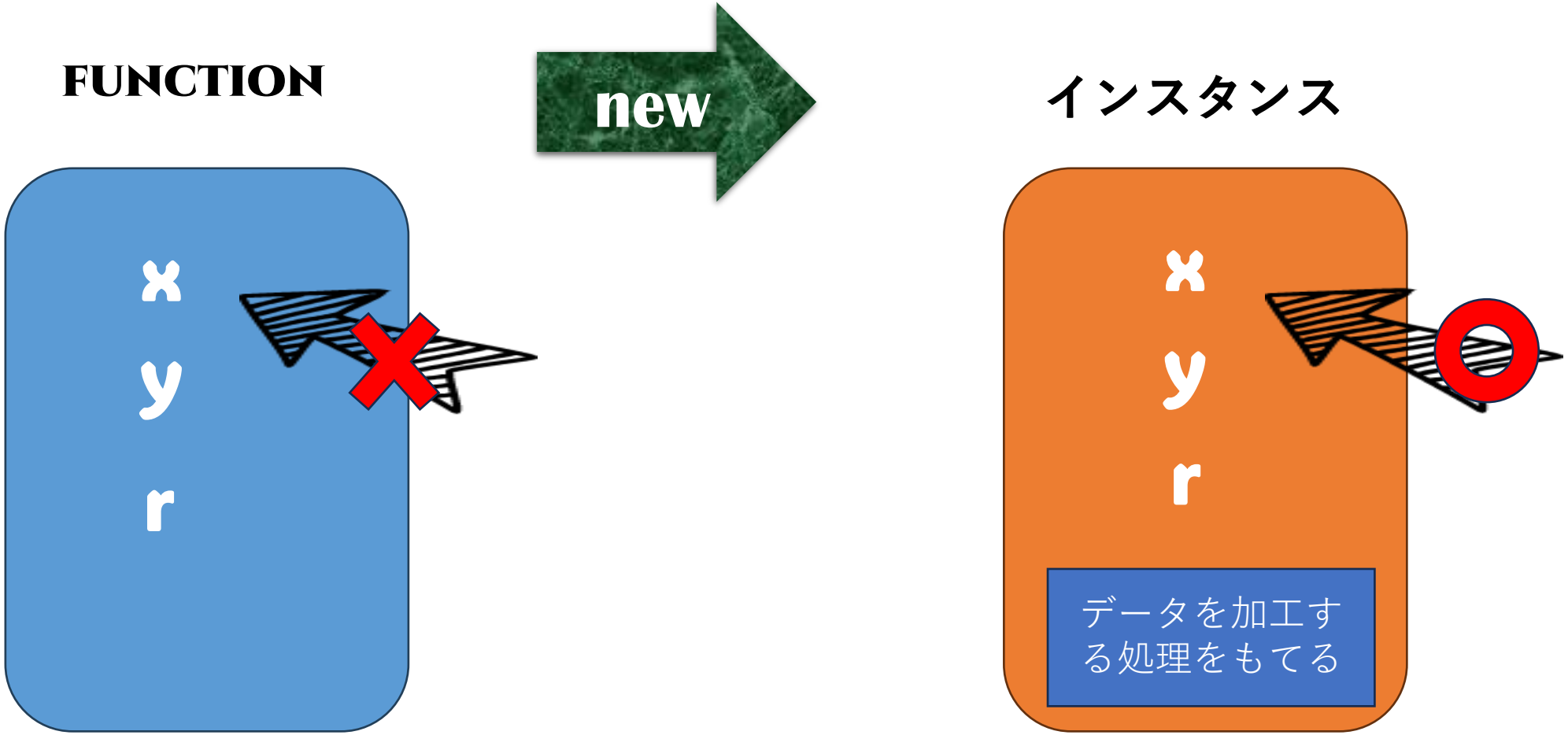
何かのひな形
どんな形か決めるモノ

FUNCTION

インスタンス

ひな形から作られたモノ
すべてがひな形の特徴を持つ

FUNCTION



データの袋の設計図みたいなもの
実際には利用できない

データを袋に入れて袋単位で
独立して取り扱えるイメージ
実際に利用できるもの

```
function Sample( a ) {
    this.a = a;  // this は自分のインスタンスの意味。
}
```

```
let sample = new Sample(10);  // インスタンス化
```

```
function Sample( a ) {
    this.a = a;  // this は自分のインスタンスの意味。
}
```

```
let sample = new Sample(10);  // インスタンス化
```

```
let a = sample.a;  // a を取り出す
```

```
sample.a = 20;  // インスタンスの a へ 20を代入
```

事前に用意しておいた クラス（Function）を使って
インスタンス化し、インスタンスからデータを取り出して
円を描きましょう。

STEP15：**配列と繰り返し**

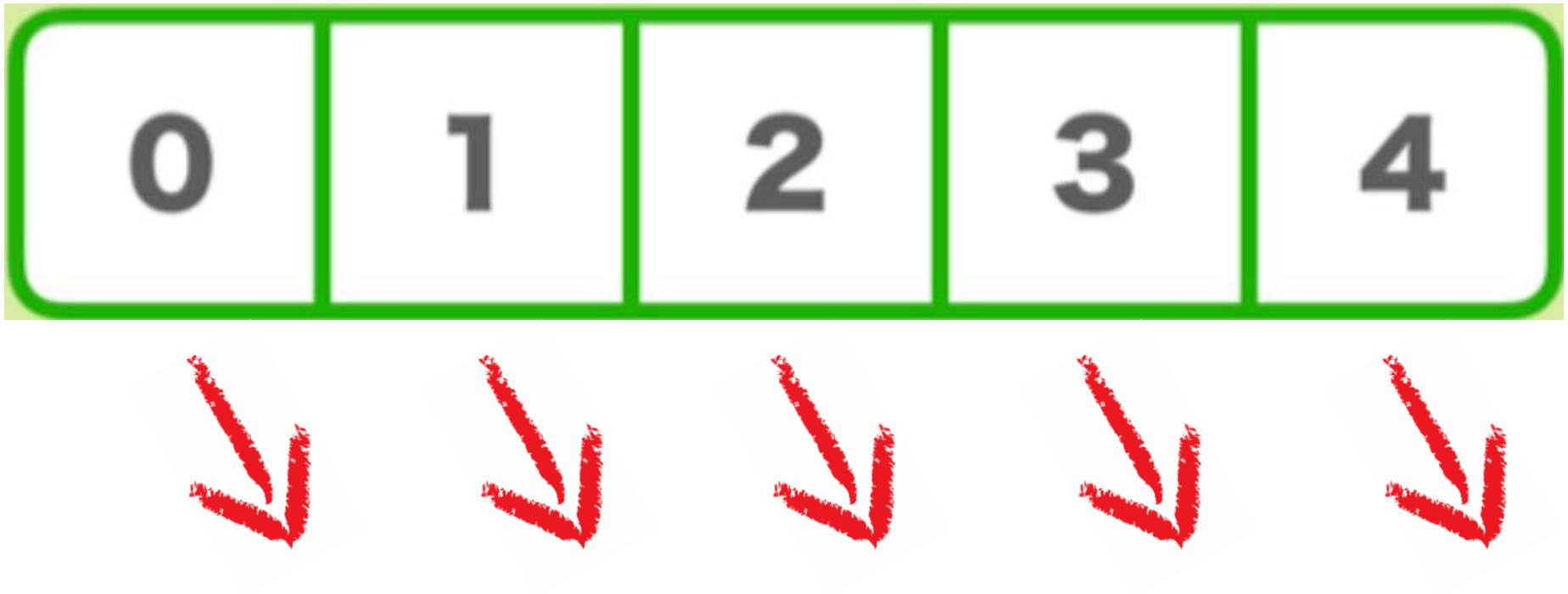
配列と繰り返しを使いこなすと
君もプログラマーの仲間入りだ

これで最後のSTEPです。
がんばろう！

配列とは？

配列：データを順番に並べた構造

変数：配列を入れておく箱



同じコードを繰り返し実行することで
順番に箱からデータを取り出したり、
データを使って何か処理をしたい！！

繰り返し処理

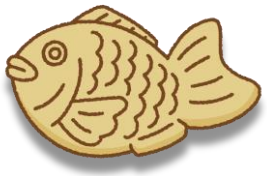
インスタンスを作って配列に入れて利用してみよう

FUNCTION

インスタンスを作る

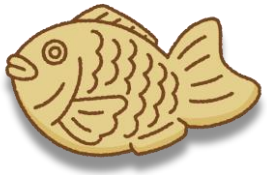
FUNCTION

new



配列へ入れる（push）

FUNCTION



PUSH

配列に入った(1番目)

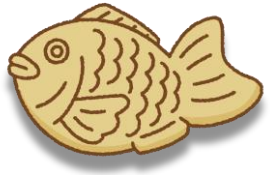
FUNCTION

インスタンスを作る

FUNCTION

配列へ入れる（push）

FUNCTION



PUSH

配列に入った(２番目)

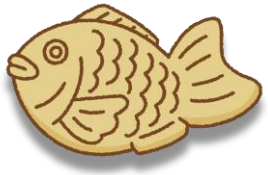
FUNCTION

インスタンスを作る

FUNCTION

配列へ入れる（push）

FUNCTION



PUSH

配列に入った(３番目)

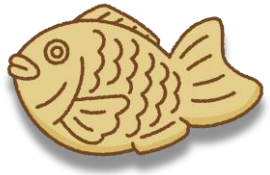
FUNCTION

インスタンスを作る

FUNCTION

配列へ入れる（push）

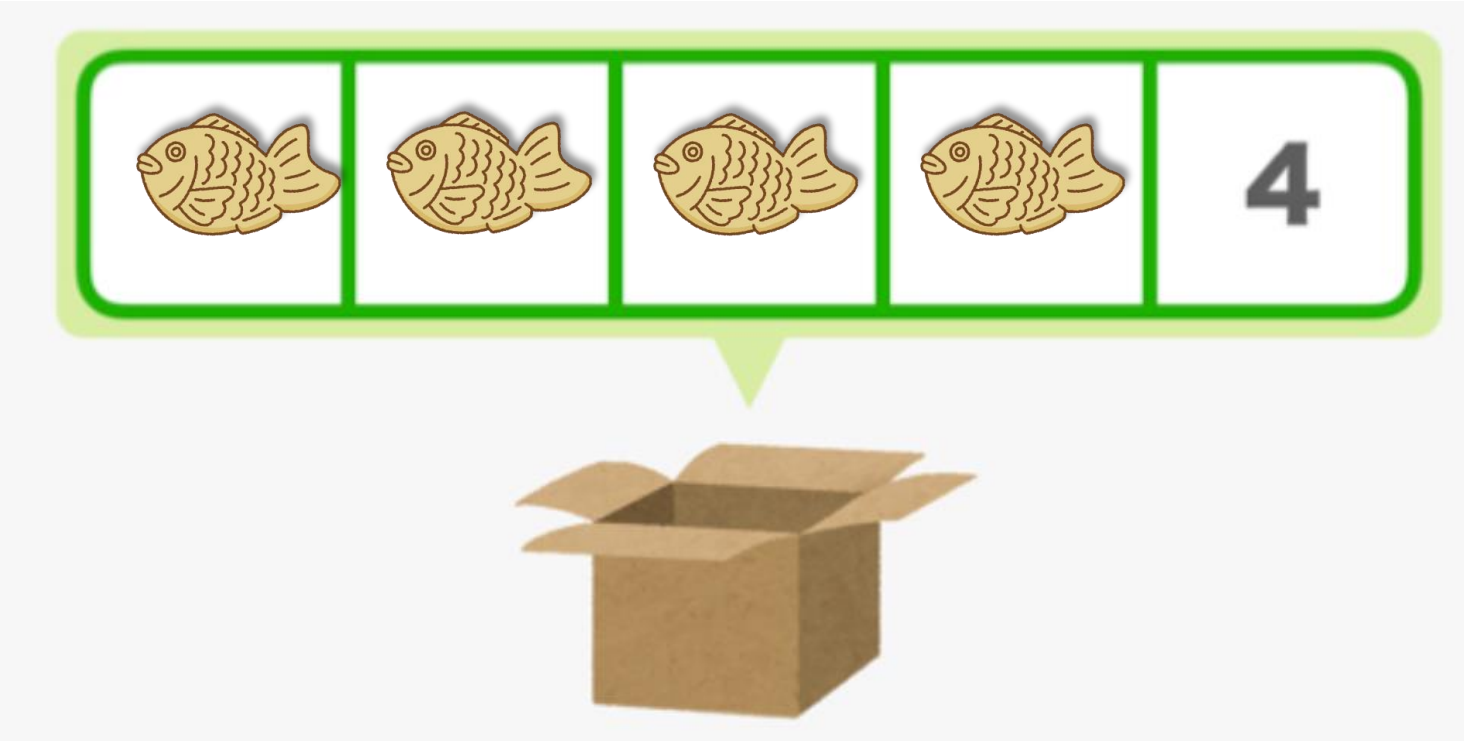
FUNCTION



PUSH

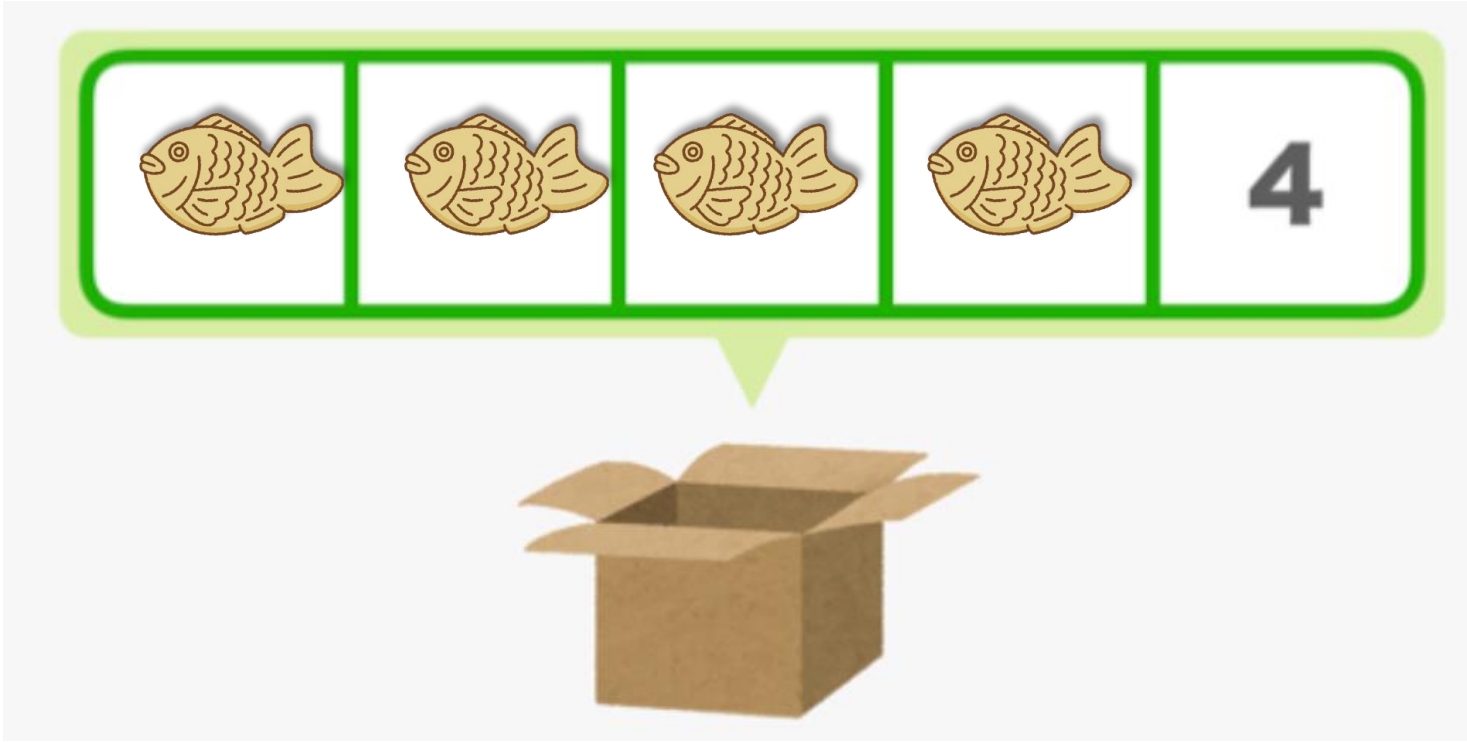
配列に入った(４番目)

FUNCTION



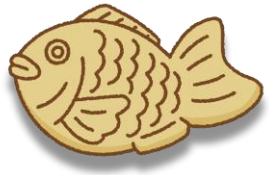
インスタンスを作る

FUNCTION

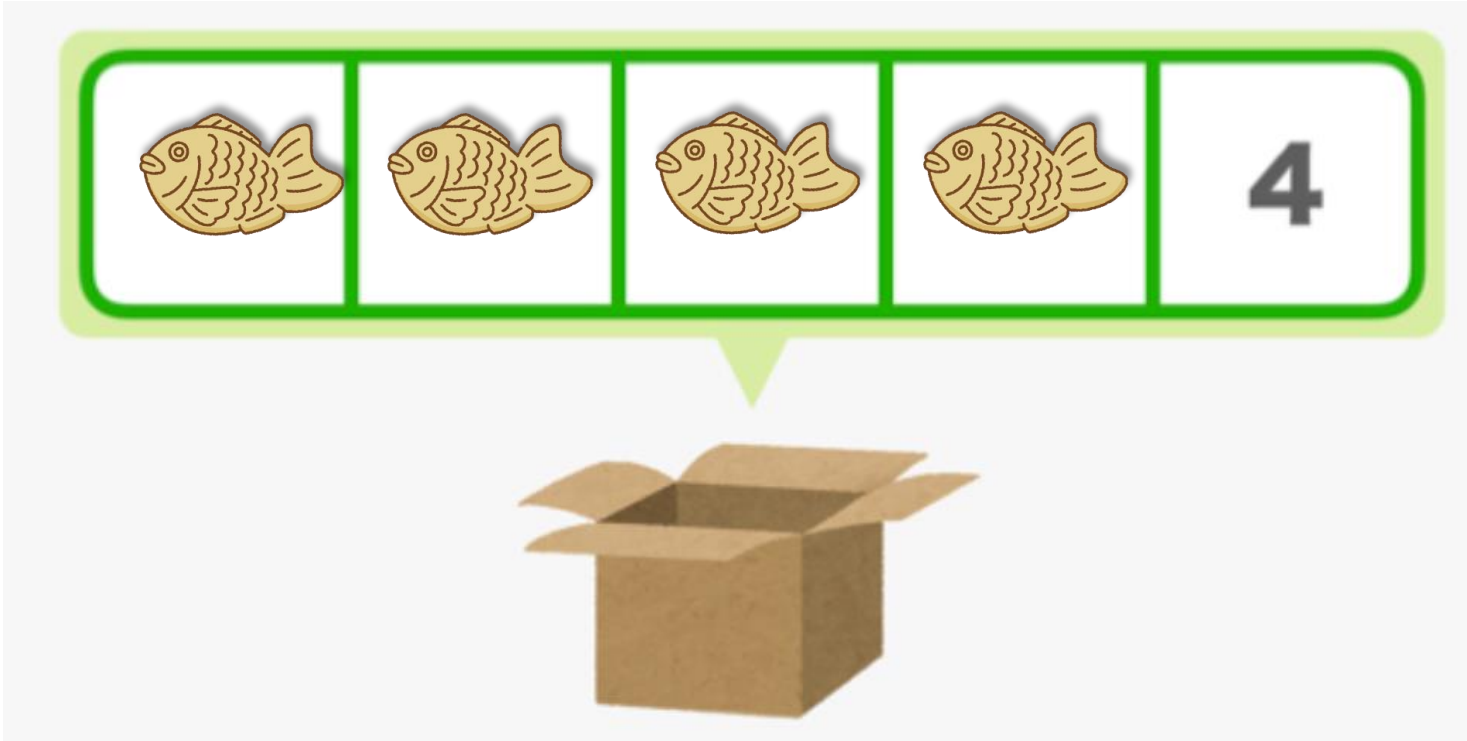


配列へ入れる（push）

FUNCTION

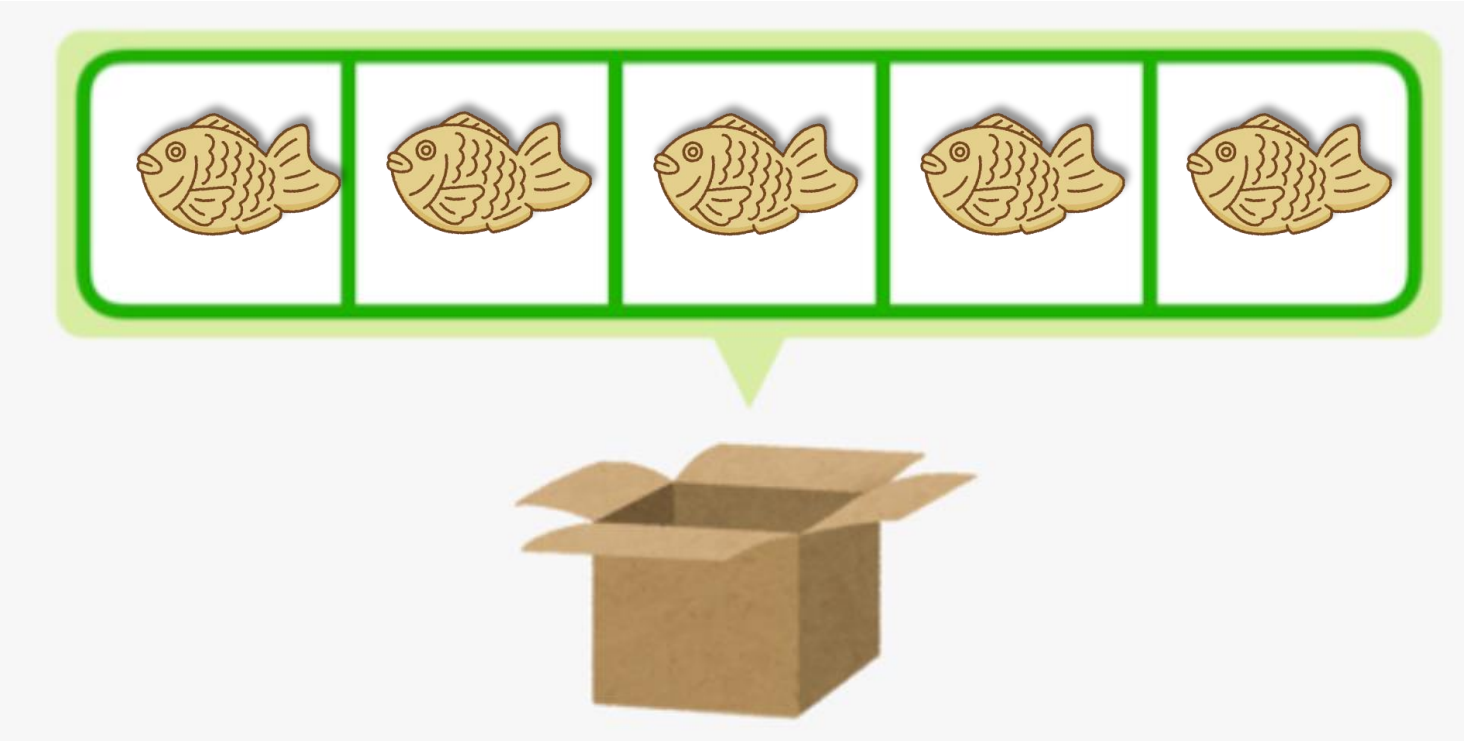


PUSH



配列に入った(５番目)

FUNCTION



①と②を **5**回繰り返す処理の書き方 **(FOREACH)**

```
Array( ５ ) . fill() . forEach {

        ① インスタンスを作る
        ② 作ったインスタンスを 配列へ push

}
```

Ball のインスタンスを
５回繰り返して作り
配列に格納しよう

Ball の定義(Function)を
書くのは大変だから、
前もって用意しておくので
利用してください。

最後までできたかな？

これであなたも
Javascriptプログラマー