ななちゃんのIT教室

マルチタートルでオブジェクトを理解しようの巻

by nara.yasuhiro@gmail.com

複数のタートルを操作する環境でななちゃんが
オブジェクト指向とは何かを勉強します

第 0.7 版　2017 年 5 月 7 日

フリー素材
http://freeillustration.net

もくじ

第1回 復習！

先生：「タートルグラッフィクスでJavaScript入門の巻」の復習からはじめましょう。

turtle.html

```
<!DOCTYPE html>
<html><body><script>
var canvas = document.createElement('canvas');
document.body.appendChild(canvas);
canvas.width  = canvas.height = 400;
canvas.style.border = "solid 2px";
var ctx = canvas.getContext('2d');
var x = y = a = 0;

function fd(step) {
  ctx.beginPath(); ctx.moveTo(200+x, 200-y);
  x += step * Math.sin(a * Math.PI / 180.0);
  y += step * Math.cos(a * Math.PI / 180.0);
  ctx.lineTo(200+x, 200-y); ctx.stroke();
}
function rt(angle) { a += angle; }
fd(100); rt(90); fd(100); rt(90);
fd(100); rt(90); fd(100); rt(90);
</script></body></html>
```

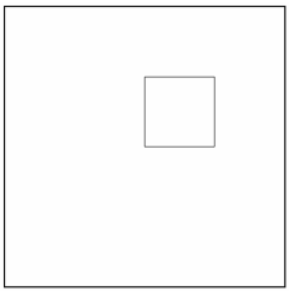
データ

fd 関数

rt 関数

毎回変わらない部分

毎回書き換える部分

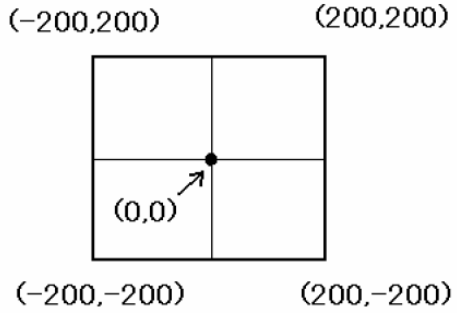


実行結果

この内容をファイルに書き込んで、ダブルクリックすると正方形が描けるのね。そして、さいごのほうの2行、fd(100) や rt(90) を含む部分を書き換えて使うということだったわね。その他の部分は、いつも同じで変化しないので、ファイルに書き込んでおいて、いつもコピーすれば良いの。

これで準備がそろったので、亀さんを何匹も使う（マルチタートル）旅に出発！

なな：レッツゴー！



タートルグラフィックスの座標系

先生：ここで使うテクニックは「オブジェクト指向」と呼ばれるもので、データとプログラムをセットにして、何組もコピーして使う方法です。

ねね：ひとつだけだったタートルをマルチタートル化するのにぴったりね！

第2回　マルチタートル！

先生：マルチタートルにすると、<script> と </script> の間の部分を、下記のように書き換えます。

```
var canvas = document.createElement('canvas');
document.body.appendChild(canvas);
canvas.width    = canvas.height = 400;
canvas.style.border = "solid 2px";
var ctx = canvas.getContext('2d');

class turtle {
    constructor(x, y, c) {
        this.x = x;
        this.y = y;
        this.c = "hsl("+c+",100%,50%)";
        this.a = 0;
    }
    fd(step) {
        ctx.strokeStyle = this.c;
        ctx.beginPath(); ctx.moveTo(200+this.x, 200-this.y);
        this.x += step * Math.sin(this.a * Math.PI / 180.0);
        this.y += step * Math.cos(this.a * Math.PI / 180.0);
        ctx.lineTo(200+this.x, 200-this.y); ctx.stroke();
    }
    rt(angle) { this.a += angle; }
}

var t1 = new turtle(-100, 100, 0);
var t2 = new turtle( 100, 100, 240);

for (var i=0; i<4; i++) {
    t1.fd(50); t2.fd(50);
    t1.rt(90); t2.rt(90);
}
```

毎回同じ部分

クラス定義

データ

fd 関数

rt 関数

亀さん誕生！

毎回書き換える部分

このプログラムでは、JavaScript の新しい機能を使っています。Internet Explorer 11 では使えません。Edge 14 以降、FireFox、Chrome ブラウザが必要です。

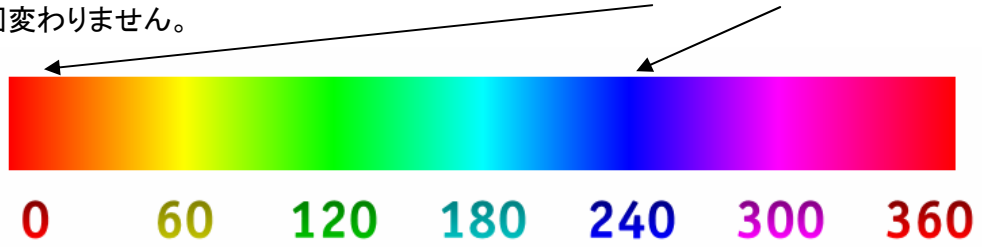
今は理解できなくて大丈夫

t1 の足跡

T2 の足跡

実行結果

ここでは、2 匹の亀さん、t1 さんと、t2 さんを使っています。
「var t1 = new turtle(-100, 100, 0);」は、画面中央から、左に 100歩、上に 100歩動いた位置に亀さん 1号を配置するということ。最後の「0」は、色を表します。0 は赤、240 は青。太字以外の部分は、毎回変わりません。



なな：ラジャー！

第3回 書き方の工夫

先生： このプログラムの

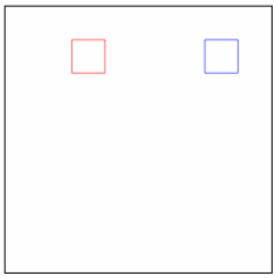
```
for (var I = 0 ; I < 4; i++ ) {
    t1.fd (50) ; t2.fd (50) ;
    t1.rt (90) ; t2.rt (90) ;
}
```

の部分を、for を使わないで書くと

```
t1.fd(50); t1.rt(90); t1.fd(50); t1.rt(90); t1.fd(50); t1.rt(90); t1.fd(50); t1.rt(90);
t2.fd(50); t2.rt(90); t2.fd(50); t2.rt(90); t2.fd(50); t2.rt(90); t2.fd(50); t2.rt(90);
```

となるけど、次のような書き方もできます。↓

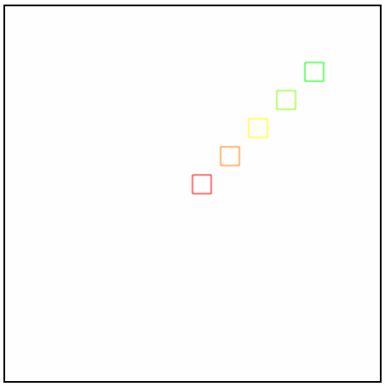
```
with ( t1 ) {
    fd (50) ; rt (90) ; fd (50) ; rt (90) ;
    fd (50) ; rt (90) ; fd (50) ; rt (90) ;
}
with ( t2 ) {
    fd (50) ; rt (90) ; fd (50) ; rt (90) ;
    fd (50) ; rt (90) ; fd (50) ; rt (90) ;
}
```



実行結果

こんなこともできます。5 匹の亀さんを使っています。

```
for (var I = 0; I < 5; i++) {
    var t = new turtle(i*30, i*30, i*30);
    for (var j=0; j<4; j++) {
        t.fd (20) ;
        t.rt (90) ;
    }
}
```



実行結果

上のプログラムだと、次のステップで、5 匹の亀さんをそれ以上使えませんが、

```
var turtles = [ ];

for (i=0; i<5; i++) {
    turtles [ i ] = new turtle(i*30, i*30, i*30) ;
    for (var j = 0; j < 4; j++) {
        t = turtles [ j ] ;
        t.fd (20) ;
        t.rt (90) ;
    }
}
```

このようにすれば、次のステップで、5 匹の亀さん、turtles [0]、turtles [1]、、、turtles [4] を turtles [0].fd (20) ; のように使うことができます。

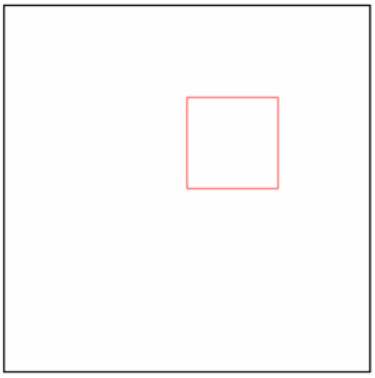
第4回　プログラムの部品化

先生：　プログラムの部品化、たとえば、正方形を描くプログラムを部品化する場合、

```
var t1 = new turtle(0, 0, 0);

function square(s) {
    t1.fd(s); t1.rt(90);
    t1.fd(s); t1.rt(90);
    t1.fd(s); t1.rt(90);
    t1.fd(s); t1.rt(90);
}

square(100);
```

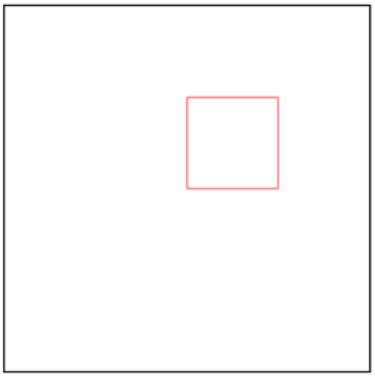


実行結果

上記のようにすると、亀さん　t1　だけ専用の部品になってしまうけど、

```
class betterTurtle extends turtle {
    square(s) {
        this.fd(s); this.rt(90);
        this.fd(s); this.rt(90);
        this.fd(s); this.rt(90);
        this.fd(s); this.rt(90);
    }
}

var t1 = new betterTurtle(0, 0, 0);
t1.square(100);
```



実行結果

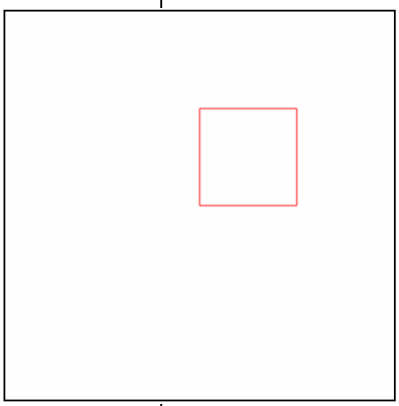
このようにすると、新しく作るすべての亀さんで使える部品になります。

第5回 書き方の工夫（連結）

先生： さらに、「変化しない部分」、「毎回使う同じ部分」の関数定義のさいごに return this; を入れると、最下段のような、連結が可能になります。

```
class turtle {
    constructor(x, y, c) {
        this.x = x;
        this.y = y;
        this.c = "hsl("+c+",100%,50%)";
        this.a = 0;
    }
    fd(step) {
        ctx.strokeStyle = this.c;
        ctx.beginPath(); ctx.moveTo(200+this.x, 200-this.y);
        this.x += step * Math.sin(this.a * Math.PI / 180.0);
        this.y += step * Math.cos(this.a * Math.PI / 180.0);
        ctx.lineTo(200+this.x, 200-this.y); ctx.stroke();
        return this;
    }
    rt(angle) { this.a += angle; return this; }
}

var t1 = new turtle(0, 0, 0);
t1.fd(100).rt(90).fd(100).rt(90).fd(100).rt(90).fd(100).rt(90);
```



先生： JavaScript のプログラムで、「 . 」が出てきたら、「オブジェクト」の機能を使っていると考えて、まず、間違いありません。

var inp = document.getElementById(“in”); とか、
var x = inp.value; とか、
inp.value = 100; とか。

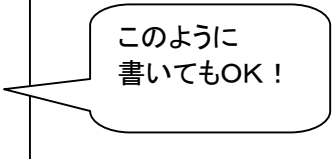
これは、Web (HTML) のページデータを、オブジェクトの形にして、JavaScript から操作できるようにする「DOM」という仕組みを使っています。DOM = Document Object Model。

Math.random(); や Math.sqrt(); は、数学処理機能を集めた「Mathクラス」の関数です。

な な： 「3.14」の「 . 」は？

先生： それは、オブジェクトとは関係ありません。

```
var t1 = new turtle(0, 0, 0);
t1.fd(100).rt(90)
   .fd(100).rt(90)
   .fd(100).rt(90)
   .fd(100).rt(90) ;
```



JavaScript の オブジェクトの使い方のまとめ

(1) 「オブジェクト」のもっとも基本的な使い方

```
var d = { a:1, b:2, c:3 };
alert(d.b);                    // 2
```

アラートで、「2」と表示されますよという意味のコメント。

a = 1; b = 2; c = 3; というのを、ひとつにまとめたようなもの。これがオブジェクト。

(2) オブジェクト内のそれぞれのデータは、数字でも、文字列でも良い。
関数定義でさえ良い。

文字列を数字に変換

```
var d = { a:1, b:2, c:function() { return Number(this.a)+Number(this.b); } }
alert(d.c() ) ;                    // 3
```

this.b は、このオブジェクト内の b という意味。

この例では、「c」が、「a」と「b」を足した結果を返すという関数。↑

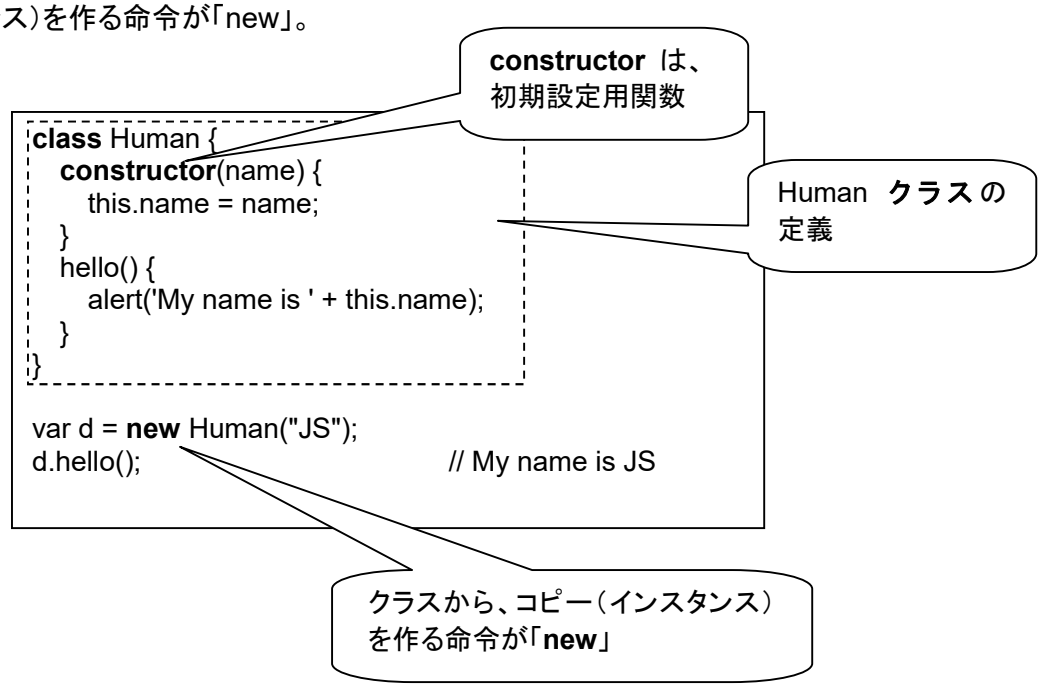
```
var d = { a:1, b:2, c:function(){return Number(this.a)+Number(this.b);} }

d.a = 3;
d.b = 5;
alert(d.c());                    // 8
```

データを書き換えることもできる。↑

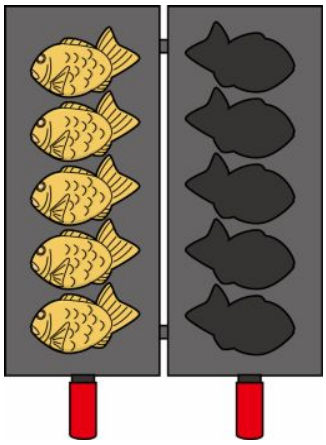
(3) オブジェクトのひな型を作って、コピーする

ひな型のことを「クラス」といいます。作ったコピーのことを「インスタンス」といいます。クラスからコピー（インスタンス）を作る命令が「new」。



この例で作られるインスタンス d の内容は、
{ name: JS, hello: function { alert('My name is ' + this.name); } です。

クラスは、たい焼きの型のイメージ。インスタンスはたい焼きのイメージです。constructor は、つぶあん、こしあん、うぐいすあんなどを使って、たい焼きを焼くというしくみで、それを呼び出すのが new というイメージです。



http://illpop.com/p_site03.htm

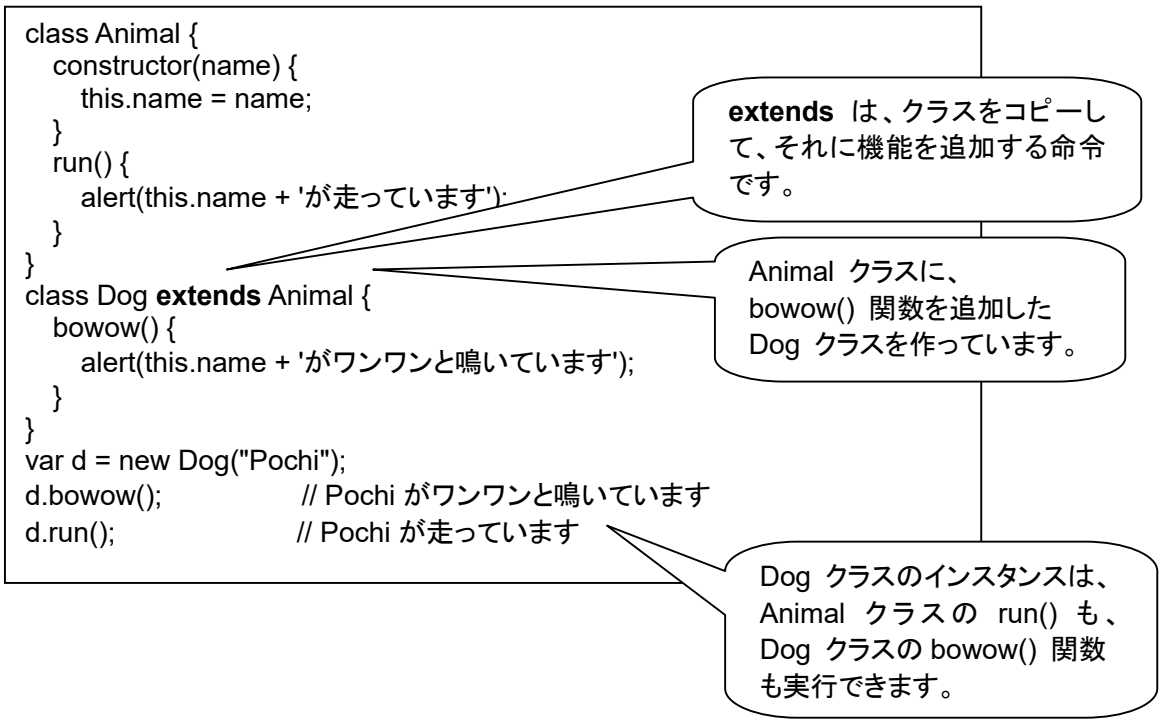
```
constructor(name) {
    this.name = name;
}
```

は、中身を指定した たい焼き型 を用意する部分、

```
var d = new Human("JS");
```

は、「つぶあん」が中身の たい焼き d を 焼く部分 という感じです。

(4) ひな型を拡張する



上の例では、extends を使って、Animal クラスをコピーして、それに機能を追加した Dog クラスを作っています。このような、「コピーして、それに機能を追加した」クラスのことを「継承クラス」といいます。コピーする元のクラスのことを「継承元クラス」といいます。

継承クラス定義で、継承元クラスと同じ関数名を使うと、上書きされますが、継承クラスの定義中で、super.関数名() を使って、継承元クラスの関数を参照できます。継承元のコンストラクタは、super() を使って参照できます。継承クラス定義の中で、継承元クラスのコンストラクタに機能を追加する場合に使います。

```
class Animal {
   constructor(name) {
      this.name = name;
   }
   run() {
      alert(this.name + 'が走っています');
   }
}
class Dog extends Animal {
   constructor(name, age) {
        super(name);
        this.age = age;
   }
   bowow() {
      alert(this.name + 'がワンワンと鳴いています');
   }
}

var d = new Dog(“Shiro”, 10);
alert(d.age) ;                          // 10
d.run();                                // Shiro が走っています
d.bowow() ;                             // Shiro がワンワンと鳴いています
```

新しい HSLカラーの使い方（CSS 3で追加されました）

色相（Hue）、彩度（Saturation）、輝度（Lightness/Luminance）の3成分

色相：色味を0～360度の範囲の角度で表す。0度は赤で、
その反対側に位置する180度は赤の反対色にあたる青緑。
反対色を見つけるのも容易。

彩度：純色から彩度が落ちる、灰色になっていくという考え方。
0% ～ 100%、灰色 ～ 鮮やかに。

輝度：輝度0%を黒、100%を白とし、その中間（50%）を純色とする。

hsl(120, 50%, 50%)
hsla(120, 50% 50%,.3)　　透明度つき