楽しいWebアニメーション

科目の概要

従来のHTMLで作成するWebページは動きませんでしたが、HTML5ではWebページ上で、標準でアニメーションを行うことができます。本講義では、そのアニメーションの仕様である「CSSアニメーション」と「canvas API」に焦点をあてて解説を行います。それらを通してHTML5によるWeb表現力の無限の可能性を体験することが目標です。

学習について

ホームページ制作やプログラミングに触れたことのない学生を想定して講義を行います。

授業中に演習の時間を設けます。制作物の出来不出来ではなく、授業を通してWebアニメーションに触れていただくことが大事ですので、気軽に、そして積極的に挑戦してください。

各回の講義内容

※ 第15回終了後に最終試験があります

教材

授業ページ

https://nakashimmer.github.io/YashimaWebAnimation/ （chromeで開くこと）

授業テキスト

https://nakashimmer.github.io/YashimaWebAnimation/webanime.pdf (印刷推奨)

JSくん

http://springgreen.jp/yashima/webanime/js/ （chromeで開くこと）

講師

中島俊治（なかしましゅんじ）

- 岡山県生まれ、1997年上京ソフトバンク入社、GeoCities JAPAN立ち上げ、Yahoo! JAPAN等のベンチャ企業の後独立。
- 現在は、大学や専門学校でHTML5を教える。マイクロソフトMVPアワード受賞(2014-)。
- HTML5講座「東京アプリ・ワークショップ」。 https://tokyoappworkshop
- Web技術振興協会(HTML5検定試験)代表理事。 https://webtechpromo.org
-

1. Webアニメーションに触れてみよう

Webアニメーション

いよいよWebアニメーションの授業のはじまりです。まずはWebアニメーションはどんなものかのサンプルを示しますので楽しみましょう。(**授業ページへ**)

- 主にCSS3によるWebアニメーション
- 主にcanvasによるWebアニメーション

HTML

HTMLはハイパテキスト・マークアップ・ランゲージの略

- ハイパーテキスト＝瞬時に関連する情報を自由自在に参照できるリンクの仕組み
- マークアップは、記号（タグ）をつけて、プログラムに理解できるようにすること

HTML5

HTMLの5回目の大幅な改訂版。2014年10月28日W3Cが勧告。

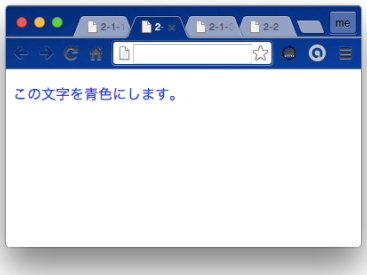
HTML5のメインの技術仕様

- 狭義HTML5(マークアップ・タグ・文書の構造を作る)
- **CSS3**（ページデザイン・要素スタイル）
- JavaScript（プログラム）
- 様々なAPI(Canvas APIほか)

2.　CSSアニメーション（CSSの基礎理解）

2.1.CSSの設置の方法

CSSの基本形は「**セレクタ{プロパティ:値;}**」の形になります。今回はstyle要素の中にCSSを記述する方法で演習を行います。

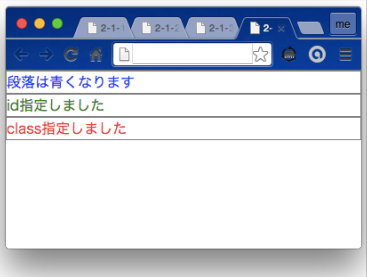



```
<style>
 p{color:blue;}
</style>
<p>この文字を青色にします。</p>
```

2.2. セレクタ

ページ上でデザインされる要素の指定をセレクタといいます、それには４つの種類があります。

- *:ユニバーサルセレクタ（ページ上の全ての要素）
- pなどのタグ：**要素セレクタ**（タグ）
- .～：**classセレクタ**（HTML内ではclass="～"に対応）
- #～：**idセレクタ**（HTML内ではid="～"に対応）




```
<style>
 *{margin:0; padding:0;}
 p{color:blue; border:1px solid gray;}
 #id1{color:green;}
 .css1{color:red;}
</style>
<p>段落は青くなります</p>
<p id="id1" class="css1">id指定しました</p>
<p class="css1">class指定しました</p>
```

3.　CSSアニメーション（CSS3のリッチデザイン）

CSSは静的なデザインを作るのが得意ですが、リッチなデザインができるようになりました。

3.1.枠線

borderプロパティは、枠線をデザインします。(1pxは線の太さ、solidは線種で実線、grayは線の色で灰色です)

```
<style>
 .box{
  width:300px; height:200px; margin:50px auto;
  line-height:200px; text-align:center;
  border:1px solid gray;
 }
</style>
<div class="box">枠線を付けます</div>
```

3.2.角丸

border-radiusプロパティは、角を丸くします。
（15pxは角丸の半径です）

```
<style>
 #box1{border-radius:15px;}
</style>
<div id="box1" class="box">角を丸くします</div>
```

3.3.影

box-shadowプロパティは、影を作ります。
（値の最初から、横方向のズレ、縦方向のズレ、ぼかし具合、影の色）

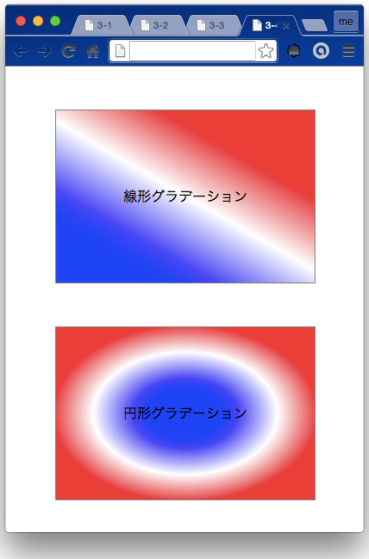
```
<style>
 #box2{box-shadow:3px 3px 3px gray;}
</style>
<div id="box2" class="box">影を作ります</div>
```

3.4.グラデーション

backgroudプロパティは、背景を指定します。その値の、linear-gradientとradial-gradientは、グラデーションです。

- linear-gradient（30degは30度の傾きで、青色が20%の位置まで、白が50%の位置で、赤が80%の位置になり、その間がグラデーションになります。）
- radial-gradient（中心から、青が20%の位置まで、白が50%、赤が80%の位置になり、その間がグラデーションになります。）

```
<style>
 #box3{background:linear-gradient(30deg,blue 20%, white 50%, red 80%);}
 #box4{background:radial-gradient(blue 20%,white 50%,red 80%);}
</style>
<div id="box3" class="box">線形グラデーション</div>
<div id="box4" class="box">円形グラデーション</div>
```



4.　CSSアニメーション（CSS3の変形）

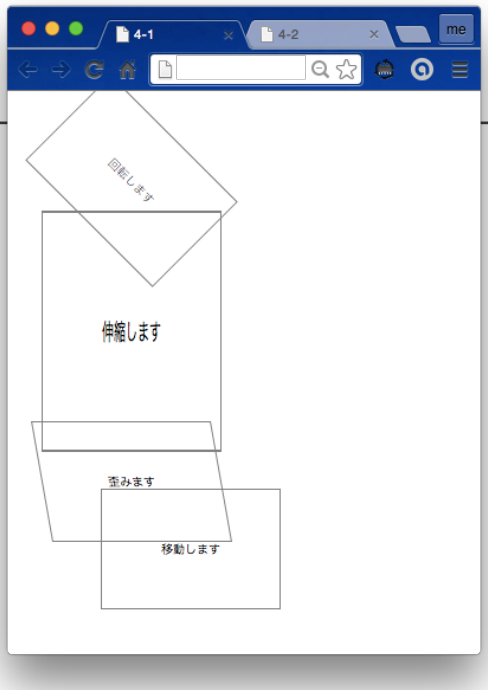
4.1.変形

CSS3では「変形」が追加されました。変形(transform)には、次の4つがあります。

- 回転(rotate：パラメータは角度)
- 伸縮(scale：パラメータは倍率)
- 歪み(skew：パラメータは横方向、縦方向の角度)
- 移動(translate：パラメータは横方向、縦方向の移動距離)

変形を組み合わせたければ、transformの値に空白を挟んで複数の変形の値を記述します。

```
<style>
 .box{
  width:300px; height:200px; margin:50px auto;
  line-height:200px; text-align:center;
  border:1px solid gray;
 }

 #box1{transform:rotate(45deg);}
 #box2{transform:scale(1,2);}
 #box3{transform:skew(10deg,0deg);}
 #box4{transform:translate(100px,-140px);}
</style>

<div id="box1" class="box">回転します</div>
<div id="box2" class="box">伸縮します</div>
<div id="box3" class="box">歪みます</div>
<div id="box4" class="box">移動します</div>
```



4.2.トランジション(遷移)

トランジションは、CSSでできるアニメーションのひとつです。

今回は変形を時間をかけて遷移（トランジション）させます。

そのためのきっかけに「:hover」を使います。:hoverはマウスが乗ったらCSSを適用させる擬似クラスというものです。

transitionプロパティは、値に「時間をかけて変化させたいCSSのプロパティ」「遷移時間」「移動のスタイル」「遅延時間」を値に持ちます。

また、「時間をかけて変化させたいCSSのプロパティ」は、変形以外に数字を値に持つCSSのほとんどが対象ですので、widthやheight、その他のCSSも遷移可能です。

```
<style>
 .box{
  width:300px; height:200px; margin:50px;
  line-height:200px; text-align:center;
  border:1px solid gray;
 }

 .box{transition:transform 1s ease-in-out 0s;}

 #box1:hover{transform:rotate(45deg);}
 #box2:hover{transform:scale(1,2);}
 #box3:hover{transform:skew(10deg,0deg);}
 #box4:hover{transform:translate(100px,100px);}

 #box5{transition:color 1s ease-in-out 0s,background-color 1s ease-in-out 0s;}
 #box5:hover{color:white;background-color:red;}
</style>

<div id="box1" class="box">回転します</div>
<div id="box2" class="box">伸縮します</div>
<div id="box3" class="box">歪みます</div>
<div id="box4" class="box">移動します</div>
<div id="box5" class="box">様々なCSSを遷移できます</div>
```

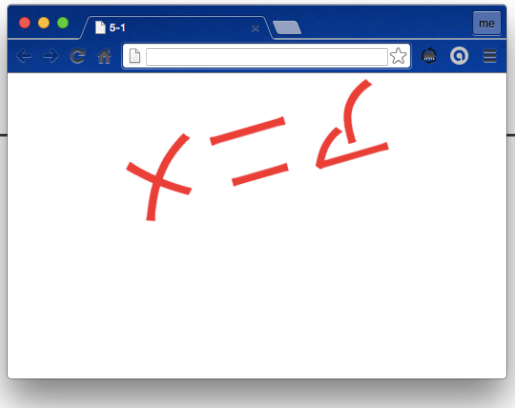
5.　CSSアニメーション（アニメーションの基礎理解）

CSSのアニメーションには、transitionとanimationがあります。
animationプロパティは、次の値でアニメーションを指定します。

- keyframesの名前
- 時間
- アニメーションのスタイル(ease-in-outはゆっくり始まりゆっくり終わる。linearは等速)
- 遅延時間
- アニメーションの繰り返し回数（infiniteは無限、1と記述すれば1回）
- 方向(normalは一方通行、alternateは往復)

5.1.CSSアニメーションの基本的な記述

```
<style>
 #box1{
  margin:100px auto;border:1px solid gray;
  font-size:100px;color:red;width:500px;text-align:center;
  animation: anime 3s ease-in-out 0s infinite normal;
 }

 @keyframes anime{
  0%{transform:rotate(0deg);}
  100%{transform:rotate(360deg);}
 }
</style>

<div id="box1">アニメ</div>
```



5.2.半透明にするには

CSSのプロパティで「opacity」というのがあります。

- 「**opacity:0;**」は透明、
- 「**opacity:0.5;**」は半透明、
- 「**opacity:1;**」は不透明です。

試してみてください。

6.　CSSアニメーション（CSS3の三次元表現）

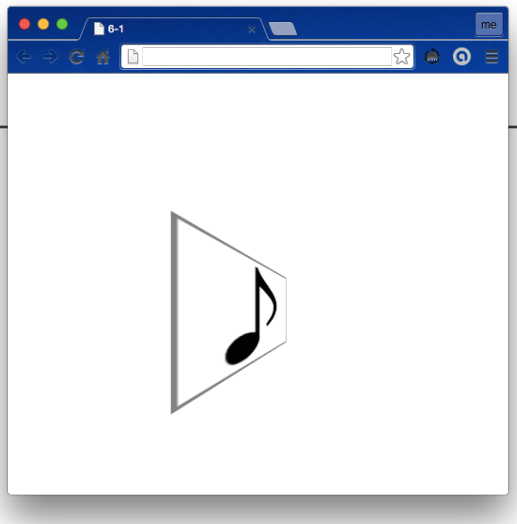
さらに三次元に挑戦しましょう。三次元にするには、三次元表示させたい要素の親の要素に対して、transform-styleプロパティで三次元を、perspectiveプロパティで奥行きを指定します。

三次元の変形には、次のようなものがあります。

- rotateX(～deg)：X軸を中心に回転
- rotateY(～deg)：Y軸を中心に回転
- rotateZ(～deg)：Z軸を中心に回転
- scaleX(～)：X軸を中心にした倍率
- scaleY(～)：Y軸を中心にした倍率
- scaleZ(～)：Z軸を中心にした倍率
- translateX(～px)：X軸での移動距離
- translateY(～px)：Y軸での移動距離
- translateZ(～px)：Z軸での移動距離

6.1.三次元表現を示します

```
<style>
 body{text-align:center;}
 .box{
  width:100px; height:100px; margin:50px auto; font-size:100px;
  line-height:100px; border:2px solid gray;
 }

 #wrap{
  transform-style:preserve-3d;
  perspective:105px;
  height:100px;margin:200px;
 }

 .box1{transform:rotateY(30deg);}

</style>

<div id="wrap">
 <div id="box1" class="box">♪</div>
</div>
```

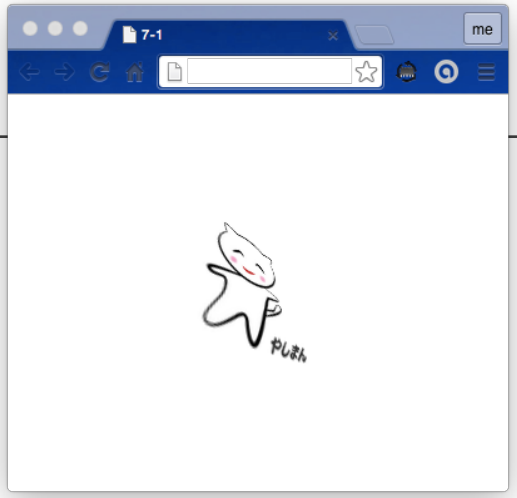


7.　CSSアニメーション（まとめ）

7.1.アニメーションの応用

では、アニメーションと三次元変形を組み合わせてみましょう

```
<style>
 body{text-align:center;}

 #wrap{
  transform-style:preserve-3d;
  perspective:105px;
  height:100px;margin:100px 0;
 }

 #box1{
  width:100px; height:100px;
  margin:0px auto;
  font-size:100px;line-height:100px;
  background-image:url(sample.png);
  background-size:100% 100%;
 }

 #box1{
  animation:animation3d 5s ease-in-out 0s infinite normal;
 }
 @keyframes animation3d{
  0%{transform:rotateX(0deg) rotateY(0deg);}
  100%{transform:rotateX(360deg) rotateY(360deg);}
 }

</style>

<div id=wrap>
 <div id="box1"></div>
</div>
```



8.　Canvas API（仕様理解）

１～７回まではCSSでのアニメーションでした。

８～１５回は「Canvas」というものの解説に入ります。

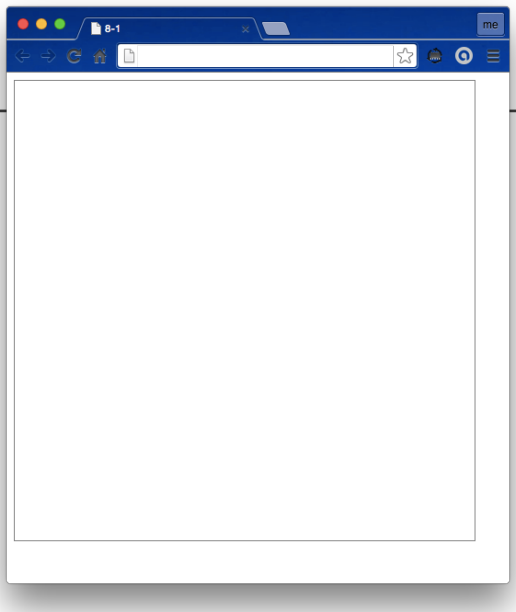
「Canvas」はHTML5の代表的な機能です。図形を描画し、それらをアニメーションとして動かすことのできるもので、ゲームなどが簡単にできます。１回めのサンプルをもう一度試しましょう。

※ここからはJavaScriptというプログラムの世界です。プログラムというと難しいと感じる人もいるでしょう。プログラミングのあまり難しいことは気にしなくても結構です。描画のしくみやアニメーションを楽しみながら、私が「ここが重要」と言ったところは覚えるようにしてください。

8.1.ファイルを作ります

canvasタグを設置します。当然ながら中身は何も表示されません。ただしcanvasタグもCSSでのデザインが可能なので、枠線や背景などを指定することも可能です。

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>はじめてのcanvas</title>
 <style>
  #mycanvas{border:1px solid gray;}
 </style>
</head>
<body>
 <canvas id="mycanvas" width="500" height="500"></canvas>

 <!-- ここにJavaScript のコードを書きましょう -->


</body>
</html>
```



9. Canvas API（JavaScript）

9.1.スクリプトを記述します

描画を扱うのはJavaScriptの役目。JavaScript内でCanvasを扱えるためのコードを記述します。そして道具箱を扱うためのコードも記述します。

- document.getElementById：対象idの要素を取得します
- .getContext("2d") ：その２次元の描画コンテキストを取得します（道具箱）

```
 <!-- ここにJavaScript のコードを書きましょう -->

<script>
 var canvas = document.getElementById("mycanvas");
 var c = canvas.getContext("2d");

 //ここに描画のソースを記述します


</script>

</body>
</html>
```

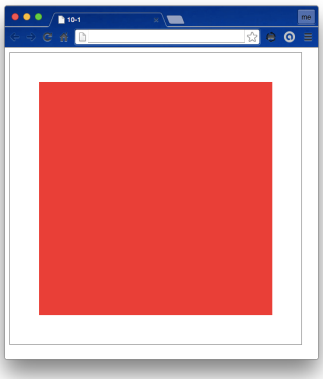
10.　Canvas API（基本図形描画１）

Canvasの準備ができたところで、基本的な図形を描画しましょう。

10.1.矩形

四角形(rectangle)です。

- .fillStyle：以降の塗り色を指定します。
- .fillRect(x座標,y座標,幅,高さ)：矩形を描画するメソッドです。

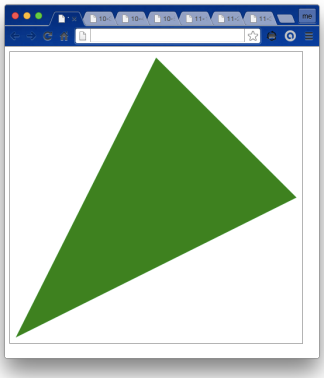


```
//矩形
c.fillStyle = "red";
c.fillRect(50,50,400,400);
```

10.2.多角形

多角形はmoveToで開始、lineToで頂点のパスを指定して塗ります。

- .beginPath()：パスの開始（パス＝軌道）
- .closePath()：パスの終了
- .moveTo(x座標,y座標)
- .lineTo(x座標,y座標)
- .fill()：塗る

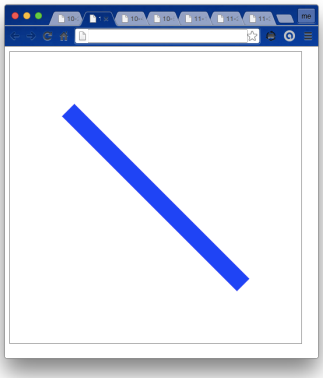


```
//多角形
c.beginPath();
c.moveTo(250,10);
c.lineTo(10,490);
c.lineTo(490,250);
c.closePath();
c.fillStyle = "green";
c.fill();
```

10.3.直線

直線は、多角形と似ています。違うのは、closePath()がないのと、パスを指定した後線を引くことです。

- .lineWidth：以降の線の幅を指定
- .strokeStyle：以降の線の色を指定
- .stroke()：線を引く

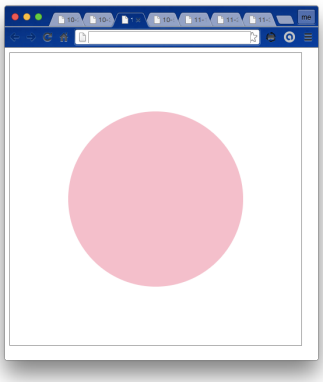


```
//直線
c.beginPath();
c.moveTo(100,100);
c.lineTo(400,400);
c.lineWidth = 30;
c.strokeStyle = "blue";
c.stroke();
```

10.4.（塗）円

Canvasにはcircleがありません。arc（円弧）を使って描画します。

- .arc(中心のx座標,y座標,半径,開始弧度,終了弧度,回転方向)
- Math.PI：円周率

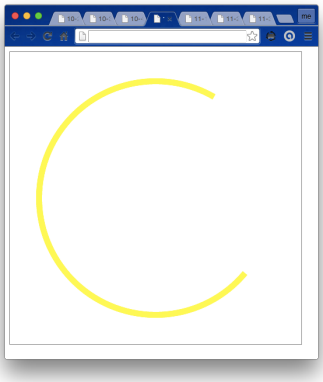


```
//塗り円
c.beginPath();
c.arc(250, 250, 150, 0, 2*Math.PI, false);
c.closePath()
c.fillStyle = "pink";
c.fill();
```

10.5.円弧（線）

その円弧では、開始弧度、終了弧度を変えることで様々な円弧を描けます。

- .arc(中心のx座標,y座標,半径,開始角,終了角,回転方向)
- 回転方向はfalse(時計回り)/true(反時計回り)
- 角度はラジアン単位
  180度→π、
  360度→2π、
  z度=z/360*2π → z/180*π



```
//円弧
var angle1 = 40/180 * Math.PI ;
var angle2 = 300/180 * Math.PI ;

c.beginPath();
c.arc(250, 250, 200, angle1, angle2, false);
c.strokeStyle = "yellow";
c.lineWidth = 10;
c.stroke();
```

11.　Canvas API（基本図形描画２）

そのほかにも曲線、グラデーション、文字なども描画できます。

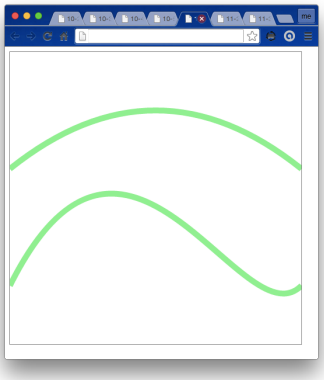
11.1.曲線

曲線には、２次の曲線と３次の曲線があります。

まずmoveToで開始点を指定し、次に次のとおりに指定します。

- .quadraticCurveTo(制御点のx座標,y座標,　終了点x座標,y座標)
- .bezierCurveTo(1つめの制御点x座標,y座標,　2つ目の制御点x座標,y座標,　終点のx座標,y座標)

```
//曲線の共通
c.strokeStyle = "lightgreen";
c.lineWidth = 10;

//二次曲線
c.beginPath();
c.moveTo(0,200);
c.quadraticCurveTo(250,0,500,200);
c.stroke();

//三次曲線
c.beginPath();
c.moveTo(0,400);
c.bezierCurveTo(200, 0, 400, 500, 500, 400);
c.stroke();
```

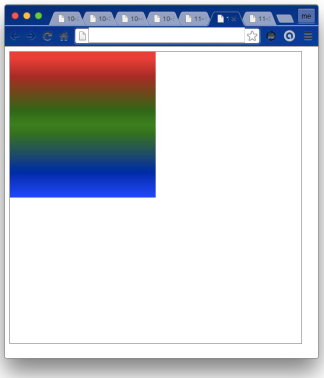


11.2.グラデーション

グラデーションを描画することができます。

まずcreateLinearGradient()でグラデーションを作り、
addColorStopで色を指定し、fillStyle（塗色）に指定します。

```
//グラデーション
var g = c.createLinearGradient(0,0, 0,250);
g.addColorStop(0,"red");
g.addColorStop(0.5,"green");
g.addColorStop(1,"blue");
c.fillStyle = g;
c.fillRect(0,0, 250,250);
```



11.3.文字

文字の描画は、fontで文字の大きさと文字フォント、fillTextで表示する文字と、座標(x,y)を決めます。。

- .font ="文字の大きさ　文字フォント"
- .fillText(文字列, x座標,y座標)：文字を表示

```
//文字
c.fillStyle = "green";
c.font = "100px serif";
c.fillText("HTML5",50,450);
```

12.　Canvas API（画像の取扱）

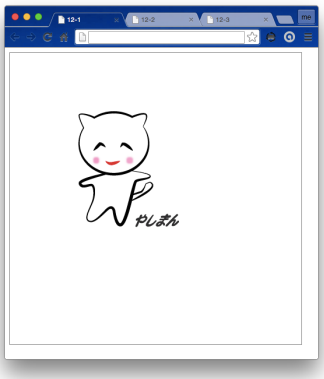
Canvasでは図形描画の他に画像を表示することができます。
好きな画像ファイルを用意してもいいでしょう。(sample.png)

12.1.画像の取り込みと表示

まず、画像をいれるものをimgという名前で作ります。
srcでファイルを指定し、drawImageで描画します。
ただし、画像の読み込みには時間がかかるので、img.onload（画像を読み込んだ後）に、drawImage()を実行します。

- .drawImage(img, x座標,y座標)：座標(x,y)にimgをそのまま表示
- 静止した状態で画像を表示させるには、imgは読み込みに時間がかかるため、画像を読み込んだ後(img.onload)に、.drawImageで図形を表示します。

```
//画像
var img = new Image();
img.src = "sample.png";
img.onload = function(){
 c.drawImage(img, 100, 100);  //※
}
```

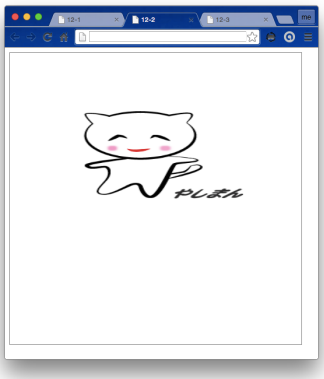


12.2.画像の大きさの変更

上記では、実寸で画像が表示されます。もし、画像の大きさを指定(300×150)にしたければ、※の部分を次のようにします。

- .drawImage(img, x座標,y座標,幅,高さ)：指定した幅、高さで拡大縮小してimgを表示

```
 c.drawImage(img, 100, 100, 300, 150);
```

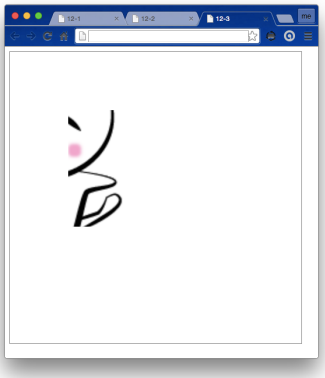


12.3.画像元の一部分を切り取って表示

さらに、元画像の(100,50)の位置から縦横100,100の大きさの画像を切り抜きたければ、※の部分を次のようにします。

- .drawImage(img, 切り出すx'座標,y'座標,幅,高さ, 描画するx座標,y座標,幅,高さ) ：imgを指定した大きさで切り出し、指定した位置・大きさで表示する

```
 c.drawImage(img, 100, 50, 100, 100, 100, 100, 200, 200);
```



13.　Canvas API（アニメーションの基礎理解）

描画の方法を理解しましたので、このセクションからはアニメーションを制作します。canvasのアニメーションは、いわゆるパラパラ漫画です。

13.1.ファイルを作りましょう

ここは今までの描画と変わりません。復習のつもりで、ソースを確認してみてください。

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>13-1</title>
 <style>
  #mycanvas{background-color:skyblue;}
 </style>
</head>

<body>
 <canvas id="mycanvas" width="500" height="500"></canvas>

 <script>
  var mycanvas = document.getElementById("mycanvas");
  var c = mycanvas.getContext("2d");
  //ここに初期化のソースを書き込みましょう



  //ここにアニメーションのソースを書き込みましょう



 </script>
</body>
</html>
```

13.2.初期値を設定する

描画するパーツの位置や色、状態の初期値を記述します。

- xは今から描画するボールのx座標、
- mukiはパラパラ漫画1コマごとの増分。1なら右に1pxずつ移動する。-1なら左に1pxずつ移動する。

```
//ここに初期化のソースを書き込みましょう
 var x = 300;
 var muki = 1;
 car count=0;
```

13.3.タイマーを設定

setInterval()は関数を50ミリ秒毎に起動させるタイマーです。もっとも重要なコードです。

- setInterval()内の関数には(丸括弧)をつけません。
- 通常は50ミリ秒がいいようです（１秒間に20コマ）
- タイマーを終了するには clearInterval(mytimer); と記述します。

```
//ここにアニメーションのソースを書き込みましょう
 var mytimer = setInterval(anime, 50);
```

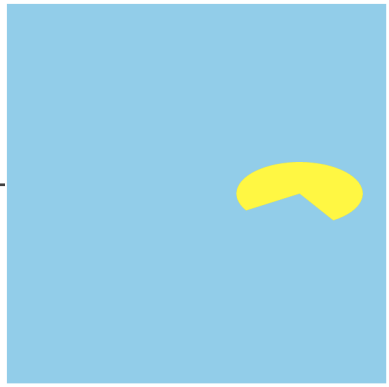
13.4.毎回の処理内容を設定

毎回毎回、①画面を消去し、②描画し、③次の状態を計算します。

function～は関数です。関数は複数の処理・動作を一つの名前にまとめたものです。ここでは「anime」という関数名に複数の処理{消去→描画→次の状態の計算}をまとめています。

- ||：「または」の記号です。シフトキーを押しながら￥マークを２回押してください。

```
  function anime(){
    //①消去----------------------------
    c.clearRect(0, 0, 500, 500);          //画面を消去

    //②状態保存＋変形----------------------
    c.save();                                 //状態の保存
    c.translate(x,300);                          //移動する
    c.scale(1,0.5);                           //伸縮する
    c.rotate(count/180*Math.PI);        //回転する

    //③(0,0)を中心としたキャラクタを描画------
    c.beginPath();
    c.moveTo(0,0);
    c.arc(0, 0, 100, 0, 1.5*Math.PI, false);
    c.closePath();
    c.fillStyle = "yellow";
    c.fill();

    //④状態復帰------------------------
    c.restore();                                 //状態の復旧

    //⑤次の状態の計算----------------------
    x = x+muki;
    if(x<100 || 400<x){muki *= -1;};
    count++;
  }
```



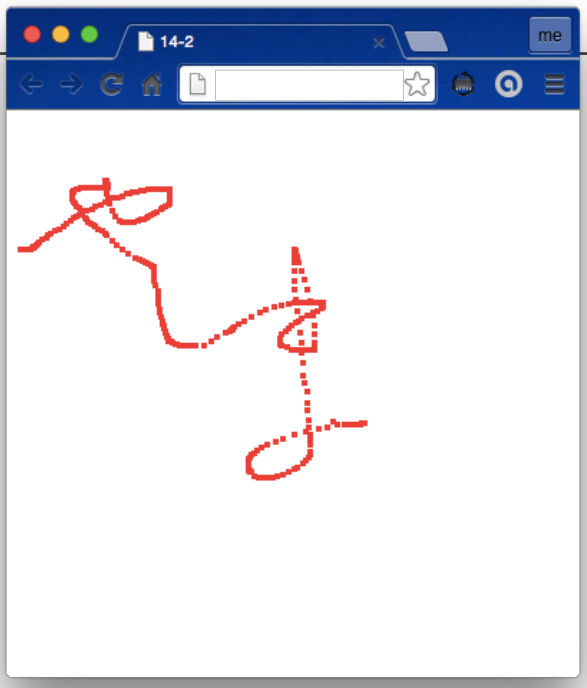
14.　Canvas API（インタラクティブ）

canvasは、マウスやタッチ、センサーと組み合わせて図形を操作し、インタラクティブなアプリケーションを作ることもできます。ここでは、マウスを利用してcanvasに小さな四角形を描画してお絵かきアプリのようなアプリケーションをご紹介します。

14.1.ファイルを作りましょう

まずは土台のファイルを作りましょう。これは今までのcanvasの作り方と全く変わりません。

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>14-1</title>
</head>
<body>
<section>
 <canvas id="mycanvas" width="500" height="500"></canvas>
</section>

<script>
 var canvas=document.getElementById("mycanvas");
 var c=canvas.getContext("2d");
 var pos={x:150,y:150};
 //ここにイベントリスナーを記述します


</script>
</body>
</html>
```



14.2.イベントリスナを記述する

マウスが動いたら、軌跡に四角形を描画するしくみは、canvasにイベントリスナを登録します。

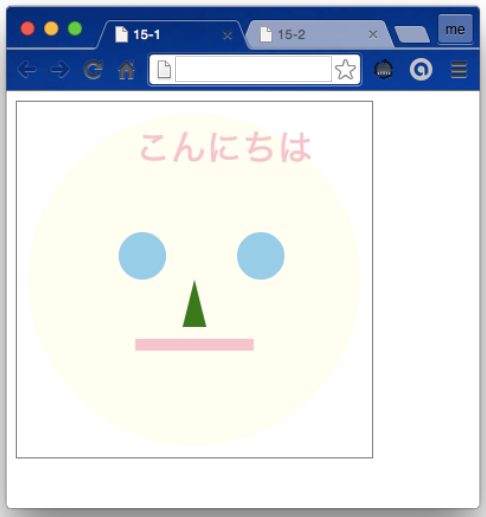
- canvas.addEventListener("イベント",動作)：なにかしらのイベントが発生したら動作する内容を定義し、canvasに追加する
- mousemove（マウスが動いている間）の他に、mousedown（クリックした瞬間）、mouseup（クリックを離した瞬間）がある
- e.clientX、e.clientY：マウスのｘｙ座標を取得する。ただしこの値はブラウザ基準の座標のためcanvas基準に補正する必要があります。それがrect.left（ｘ方向の補正値）、rect.top（ｙ方向の補正値）。rectは、 e.target.getBoundingClientRect(); によりcanvasの矩形状態の位置の値を得ている。
- function(e){}：は無名関数というもの、わざわざ関数名をつけるまでもないときに処理内容を記述する。使い捨て関数ともいわれる。eは、イベントが発生した時にイベントの情報は関数の第１引数に入ることが決まっているので、引数を任意にeとしている

```
 //ここにイベントリスナーを記述します

canvas.addEventListener("mousemove",function(e){

 var rect = e.target.getBoundingClientRect();
 pos.x = e.clientX - rect.left;
 pos.y = e.clientY - rect.top;

 c.fillStyle="red";
 c.fillRect(pos.x-2,pos.y-2,4,4);

});
```

15.　Canvas API（関数化で記述を楽に）

図形のコードを毎回書くのは面倒です。関数化して、記述を簡単にしましょう。
sample15.jsに独自関数を記述して、メインのプログラムをシンプルにしています。

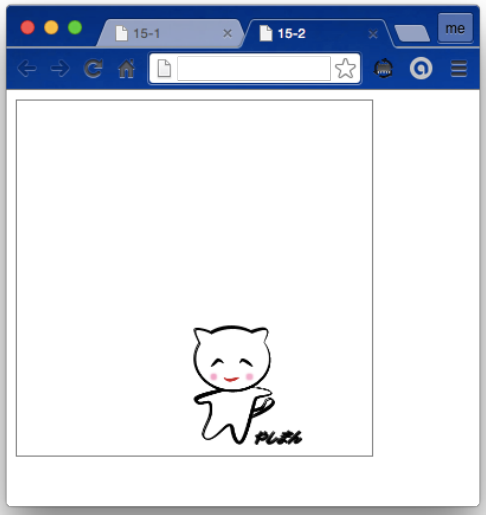
15.1.描画によるアニメーション。




```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>15-1</title>
 <style>
  canvas{border:1px solid gray;}
 </style>
</head>
<body>
 <canvas id="mycanvas" width="500"
height="500"></canvas>
 <script src="sample15.js"></script>
<script>
 //メインプログラム
 setInterval(anime,50);

//初期化
var eye = {x:150, y:130, dx:1};

 function anime(){
  frame++;
  //①画面クリア
  c.clearRect(0,0, canvas.width, canvas.height);

  //②描画
  maru(150,150,140,"#ffe");                    //輪郭
  maru(eye.x-50, eye.y, 30, "skyblue");        //左目
  maru(eye.x+50, eye.y, 30, "skyblue");        //右目
  sankaku(150, 150,140,190,160,190,"green"); //鼻
  shikaku(100,200,100,10,"pink");              //口
  msg(100,50,30,"こんにちは","black");                    //メッセージ

  //③計算
  eye.x += eye.dx;
  if(eye.x<120 || 180<eye.x){eye.dx *= -1;}
 }
</script>
</body>
</html>
```

15.2.画像の移動によるアニメーション

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>15-2</title>
 <style>
  canvas{border:1px solid gray;}
 </style>
</head>

<body>
 <canvas id="mycanvas" width="500" height="500"></canvas>
 <script src="sample15.js"></script>

<script>

 //メインプログラム
 setInterval(anime,50);

 //初期化
 var yashiman = {x:0, y:0};

 var img=new Image();
 img.src="sample.png";

 function anime(){

  frame++;

  //①画面クリア
  c.clearRect(0,0, canvas.width, canvas.height);

  //②描画+計算
  ugoku(0, 50,0,1);
  ugoku(50,200,1,1);
  ugoku(200,400,1,-1);
 }
</script>
</body>
</html>
```



15.3.独自関数のjavaScriptファイル sample15.js

```
//アニメ準備
 var canvas = document.getElementById("mycanvas");
 var c = canvas.getContext("2d");

//初期値の設定
 var frame=0;

//独自関数を定義

//①四角（矩形）
 function shikaku(x,y,w,h,color){
  c.fillStyle = color;
  c.fillRect(x,y,w,h);
 }

//②円
 function maru(x,y,r,color){
  c.beginPath();
  c.arc(x,y,r,0,2*Math.PI,false);
  c.fillStyle=color;
  c.fill();
 }

//③多角形
 function sankaku(x,y,x1,y1,x2,y2,color){
  c.beginPath();
  c.moveTo(x,y);
  c.lineTo(x1,y1);
  c.lineTo(x2,y2);
  c.closePath();
  c.fillStyle=color;
  c.fill();
 }

//④直線
 function sen(x,y,x1,y1,color){
  c.beginPath();
  c.moveTo(x,y);
  c.lineTo(x1,y1);
  c.strokeStyle=color;
  c.stroke();
 }

//⑤文字
 function msg(x,y,size,str,color){
  c.fillStyle=color;
```

```
  c.font=size+"px arial";
  c.fillText(str,x,y);
 }

//「動く」
 function ugoku(start,end,dx,dy){
  if(start<=frame && frame<=end){
    yashiman.x += dx;
    yashiman.y += dy;
  }
  c.drawImage(img,yashiman.x,yashiman.y,100,100);
 }
```