Canvas in HTML5

1. Android App with HTML5
2. Canvas
    - Template
    - Poly-line
        - triangle
        - Circle
        - Cardiac
    - Arc
    - Canvas Animation
        - Animation Structure
        - simple example
        - clock
3. Sprite animation

HTML5 gym

Default browser in Android (less than 4.4) uses "webkit" engine which is as same as the one in google chrome. In other words, Google chrome is the best choice to use to test HTML5 codes we had written.

However, the Android browser does not support all the new HTML5 functions; something would be wrong while testing HTML5 codes. Use

tools -> JavaScript Console or "Shift + Control + j" to activate the debug function.

Android (>4.4 KitKat): Progress about Webview

From this release, Chromium 30 is the web engine for the WebView native widget which, owns:

- Support for remote debugging;
- Support for new HTML5 features: Web Sockets, Web worker, IndexedDB, Animation Timing API, CSS3 Flexbox etc;
- Better performance.

But still not support: **WebGL, WebRTC, WebAudio, FullScreen and Form validation**.

Android App with HTML

Steps

- enable internet permission;
- create a webView;

- copy HTML with library into projects
- Java staff for HTML working within webView

1. internet permission

Create project, BMIapp for instance. Add "permission" in Manifest.xml in the top directory:

```
...
<uses-permission android:name="android.permission.INTERNET"/>
...
       <activity
            android:name="com.life.bmiapp.MainActivity"
            android:label="@string/app_name" >
            ...
        </activity>
```

2. WebView Creation Add a view (display), called webview01, in "main.xml" in "res/layout" directory:

```
...
<webview android:id="@+id/webview1"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
</webview>
```

3. Add HTML codes

copy all the files into the directory **$Project/src/main/assest**

4. Java Codes

```
...
package com.life.bmiapp;
    ...
import android.webkit.WebView;
    ...
public class WebviewActivity extends Activity {
       private WebView webview;
       @Override
       public void onCreate(Bundle savedInstanceState) {
              ...
           webview = (WebView)this.findViewById(R.id.webview1);
           webview.getSettings().setJavaScriptEnabled(true);
           webview.loadUrl("file:///android_asset/index.html");
       }
       ...
}
```

5. Run App

Done.

Self Practice

Try to make a html-based app.

In [ ]:

```
```

Canvas

New feature of HTML5 for creating picture online is the best choice to develop app with mobile webview.

Usage

- create a *canvas* tag with size, width and height;
- implement canvas via Javascript script.

Template

```
<script type="text/javascript">
window.onload = function {
   var a_canvas = $("a_canvas");>
   var context = a_canvas.getContext("2d");
   ...
}
</script>
<body>
<canvas id="a_canvas" style="width:100%; height:100%"></canvas>
</body>
```

This creates a canvas as follows:

Poly-line

make a poly-line, (x0,y0) to (x1,y1), and so on:

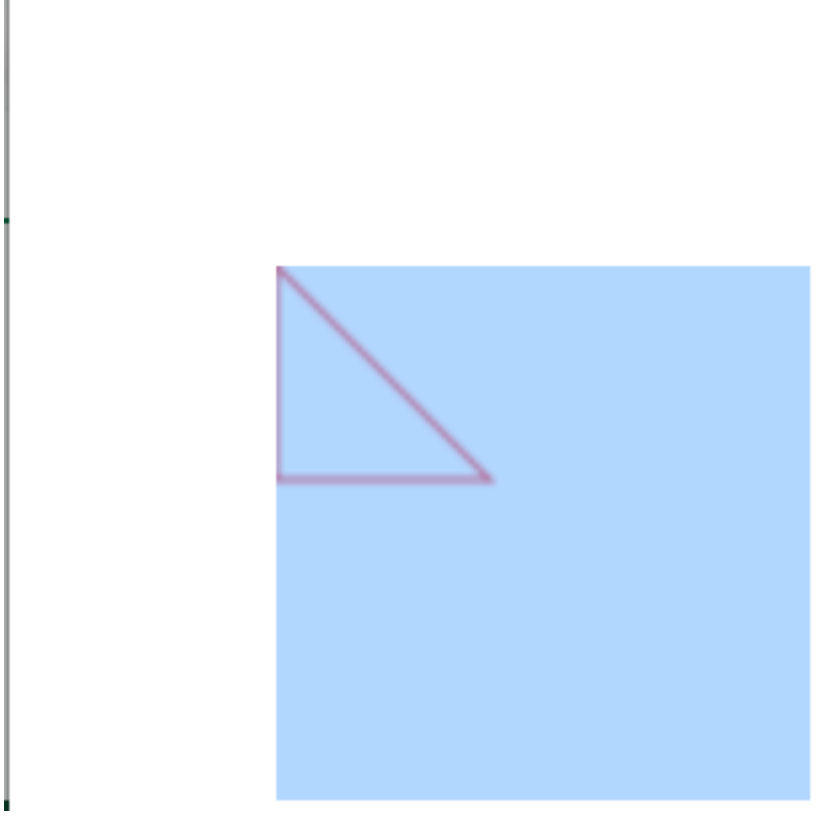
```
context.beginPath();
context.moveTo(x0,y0)
context.lineTo(x1,y1);
...
context.closePath();
context.strokeStyle="red";
context.stroke();
```

Simple Pictures

Triangle

```
<html>
  <head>
    <title>Rectangle</title>
    <script type="text/javascript">
      window.onload = function() {
         var c = document.getElementById("a_canvas");
         var context = c.getContext("2d");

         context.beginPath();
         context.moveTo(0,40)
         context.lineTo(40,40);
         context.lineTo(0,0);
         context.closePath();
         context.strokeStyle="red";
         context.stroke();
      }
  </script>
  </head>
  <body>
  <div style="position: absolute; top: 50px; left:50px;">
  <canvas id="a_canvas" width="100" height="100">
    Your browser does not support HTML5 Canvas.
  </canvas>
  </div>
  </body>
</html>
```

Result picture



Circle

$(x, y) = (r\cos t, r\sin t)$

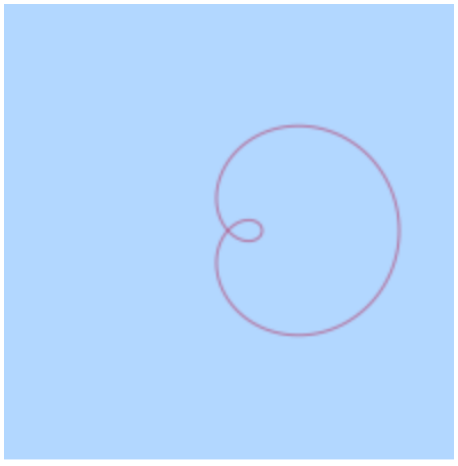
```
var dt =1;
var pi = Math.PI;
context.moveTo(230,200)
for (var i=0; i<360+1; i++){
    var t = -i*dt*pi/180
    var x= 200+30*(Math.cos(t));
    var y= 200+30*(Math.sin(t));
    context.lineTo(x,y);
}
```

Cardiac

$r = 30 + 45\cos t$ and centred at $(x, y) = (100, 100)$:

```
<script type="text/javascript">
   window.onload = function (){
      var canvas=document.getElementById("a_canvas");
      context = a_canvas.getContext("2d");
      context.beginPath();
      var dt =1;
      var pi = Math.PI;
      context.moveTo(175,100)
      for (var i=0; i<=360; i++){
          var t = i*dt*pi/180
          var x= 100+(30+45*Math.cos(t))*(Math.cos(t));
          var y= 100+(30+45*Math.cos(t))*(Math.sin(t));
          context.lineTo(x,y);
  }
  context.closePath();
      context.strokeStyle="red";
      context.stroke();
     }
</script>

<body>
 <canvas id="a_canvas" width="200" height="200"></canvas>
</body>
```

Result Picture



Self-Practice

Make a equilateral triangle by **canvas** tag.

Arc For Curves

context.arc(x, y, radius, startAngle, endAngle, anticlockwise)

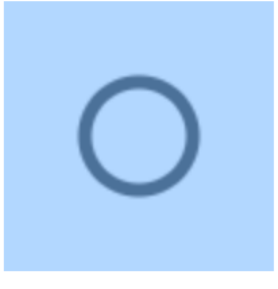
```
    context.beginPath();
    context.strokeStyle = "black";
    context.lineWidth = 5;
    context.arc(50, 50, 20, (Math.PI/180)*0, (Math.PI/180)*360, false);
    context.stroke();
    context.closePath();
```

Circle again

```
<script type="text/javascript">
      window.onload = function() {
         var c = document.getElementById("a_canvas");
         var context = c.getContext("2d");

         context.beginPath();
         context.strokeStyle = "black";
         context.lineWidth = 5;
         context.arc(50, 50, 20, (Math.PI/180)*0, (Math.PI/180)*360, fals
e);
         context.stroke();
         context.closePath();
      }
</script>
```

```
<body>
  <div style="position: absolute; top: 50px; left:50px;">
  <canvas id="a_canvas" width="100" height="100">
    Your browser does not support HTML5 Canvas.
  </canvas>
  </div>
</body>
```

Result Picture



Canvas Animation

What is Animation? A picture flow plays or changes by millisseconds.

Animation

What is Animation? Play A Sequence of pictures one by one and changes by millisseconds (about 16 frames per second (*fps*) ~ 30 *fps* at better).

Howto: Now most browsers provide efficient web technique, called *requireAnimationFrame*, for web developers to implemnet animation work on the fly.

Animation Structure

Briefly,

```
init() -> animate() -> draw()
            set up
          reqAnimFrame
```

Complete Code

```
<body>
   <canvas id="myCanvas" width="578" height="200"></canvas>
   <script>
      function init() {
            ### Basic setting for size of picture, line width, doing
animation etc.
      }

      function animate() {
         reqAnimFrame = window.requestAnimationFrame ||
         window.mozRequestAnimationFrame    ||
         window.msRequestAnimationFrame     ||
         window.oRequestAnimationFrame;
         reqAnimFrame(animate);

         draw();
      }

      function draw() {
         ...make picture ...
      }

      init();
      animate();
</script>
</body>
```

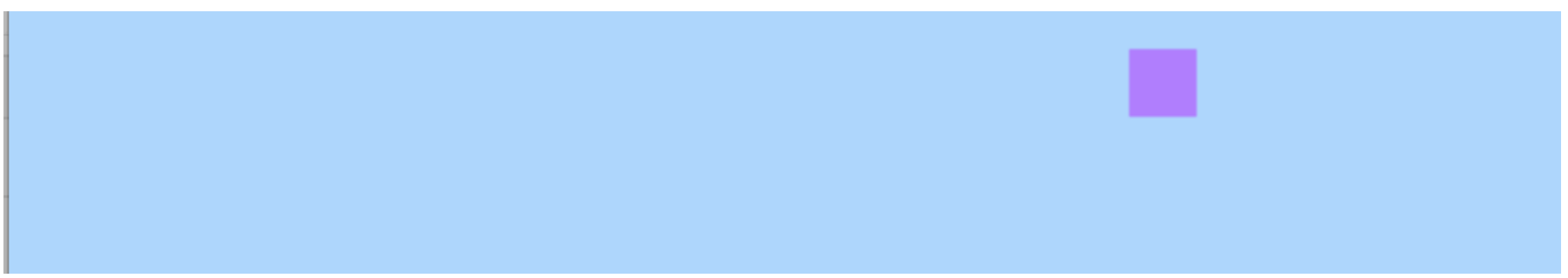
Note

The API for Canvas animation rendering is not the same for different kinds of browsers, it could overcome by re-define the reqAnimFrame to unify the commands with respect to different browers:

```
function animate() {
   reqAnimFrame = window.requestAnimationFrame  ||
     window.requestAnimationFrame ||
     window.mozRequestAnimationFrame ||
     window.msRequestAnimationFrame  ||
     window.oRequestAnimationFrame;
   reqAnimFrame( animate );
   draw();
}
```

Moving within Interval

```
<body>
  <canvas id="myCanvas" width="578" height="100"></canvas>
  <script>
    var canvas,context;
    var x =  0;
    var y = 15;
    var speed = 5;
    function init() {
       canvas  = document.getElementById("myCanvas");
       context = canvas.getContext("2d");
    }

    function animate() {

      reqAnimFrame = window.requestAnimationFrame       ||
         window.requestAnimationFrame ||
         window.mozRequestAnimationFrame    ||
         window.msRequestAnimationFrame     ||
         window.oRequestAnimationFrame;
      reqAnimFrame(animate);

      draw();
    }

    function draw() {
      x += speed;

      if(x <= 0 || x >= canvas.width-25){
        speed = -speed;
      }

      context.clearRect(0, 0, canvas.width, canvas.height);
      context.fillStyle = "#ff00ff";
      context.fillRect(x, y, 25, 25);
    }
    init();
    animate();
  </script>
```



Clock

Take a look at the snapshoot

Description

1. clock setting
    - initialisation: maker style, position etc:

```
function clock() {
   ...
   ctx.save();
   ctx.clearRect(0,0,150,150);
   ctx.translate(75,75);
   ctx.scale(0.4,0.4);
   // start from top north
   ctx.rotate(-Math.PI/2);
   ctx.strokeStyle = "black";
   ctx.fillStyle = "white";
   ctx.lineWidth = 8;
   ctx.lineCap = "round";
   ...
 }
```

- Hour markers, bold lines:

```
ctx.save();
for (var i=0;i<12;i++){
    ctx.beginPath();
    ctx.rotate(Math.PI/6);
    ctx.moveTo(100,0);
    ctx.lineTo(120,0);
    ctx.stroke();
}
ctx.restore();
```

- Minute markers, light lines:

```
ctx.save();
ctx.lineWidth = 5;
for (i=0;i<60;i++){
    // exclude hour's markers
    if (i%5!=0) {
        ctx.beginPath();
        ctx.moveTo(117,0);
        ctx.lineTo(120,0);
        ctx.stroke();
    }
    ctx.rotate(Math.PI/30);
}
ctx.restore();
```

2. Time calculation

- get time from box

```
var now = new Date();
```

- hour timer,

```
...
var hr  = now.getHours();
hr = hr>=12 ? hr-12 : hr;
...
ctx.save();
ctx.rotate( hr*(Math.PI/6) + (Math.PI/360)*min + (Math.PI
/21600)*sec )
ctx.lineWidth = 14;
ctx.beginPath();
ctx.moveTo(-20,0);
ctx.lineTo(80,0);
ctx.stroke();
ctx.restore();
```

    - minute timer,

```
...
var min = now.getMinutes();
...
ctx.save();
ctx.rotate( (Math.PI/30)*min + (Math.PI/1800)*sec )
ctx.lineWidth = 10;
ctx.beginPath();
ctx.moveTo(-28,0);
ctx.lineTo(112,0);
ctx.stroke();
ctx.restore();
```

- second timer,

```
...
var sec = now.getSeconds();
...
// Write seconds
ctx.save();
ctx.rotate(sec * Math.PI/30);
ctx.strokeStyle = "#D40000";
ctx.fillStyle = "#D40000";
ctx.lineWidth = 6;
ctx.beginPath();
ctx.moveTo(-30,0);
ctx.lineTo(83,0);
ctx.stroke();

// centered hinge
ctx.beginPath();
ctx.arc(0,0,10,0,Math.PI*2,true);
ctx.fill();
// the end hinge
ctx.beginPath();
ctx.arc(95,0,10,0,Math.PI*2,true);
ctx.stroke();
// make a empty ring at the end hinge
ctx.fillStyle = "rgba(0,0,0,0)";
ctx.arc(0,0,3,0,Math.PI*2,true);
ctx.fill();
ctx.restore();
```

- the clock,

```
            ctx.beginPath();
            ctx.lineWidth = 14;
            ctx.strokeStyle = '#325FA2';
            ctx.arc(0,0,142,0,Math.PI*2,true);
            ctx.stroke();

            ctx.restore();
```

3. initialisation

```
        function init(){
            animate();
        }
```

4. make animation

```
    function animate() {
        reqAnimFrame = window.requestAnimationFrame       ||
        window.requestAnimationFrame ||
        window.mozRequestAnimationFrame    ||
        window.msRequestAnimationFrame     ||
        window.oRequestAnimationFrame;
        reqAnimFrame( animate );
        clock();
    }
```

SpriteAnimation

Another traditional animation, just play the frame1, clear display, play frame2, clear, and so on.

Additionally, all make a similar spites but move to the right.

rendering api

```
ctx.drawImage(image, Src_x, Src_y, Src_Width, Src_Height,
                     Dest_x, Dest_y, Dest_Width, Dest_Height);
```

```
|-------------------------------------------------
|        :                :
|        : Src_y          :
| Src_x  :  Src_Width     :
|........|------------|   :
|        |            |Src_Height
|        |  Image     |   :
|        |____________|   : Dest_y
|                         :
|                         :   Dest_Width
|.........................|-----------|
|        Dest_x           |           | Dest_Height
|                         |  Image    |
|                         |___________|
|
```

In [1]:

```
from IPython.display import HTML
```

In [2]:

```
canvas="""
<canvas id="myCanvas" width="500" height="100"></canvas>
"""
```

In [9]:

```
JSCode="""
<script>
    var speed = 25;
      var frames = 4;
      var currentFrame = 0;
      var Src_Width=100;
      var Src_Height=100;
      var speed = -10;
      var f_width=400;

      canvas = document.getElementById("myCanvas");
      var width = canvas.width;

      var height = canvas.height;
      ctx = canvas.getContext("2d");
      // Moving to left
      image = new Image()
      image.src = 'imgs/sprite.png';
      //movin to right
      image2 = new Image()
      image2.src = 'imgs/sprite2.png';
      var draw = function(){
          f_width += speed;

          // change speed if hits wall and the position of x
          if(f_width <= 0-25 || f_width >= width-25){
             speed = -speed;
          }
          ctx.clearRect(0, 0, width, height);
          // change direction while hits wall
          if(speed <= 0){
             ctx.drawImage(image, 0, height * currentFrame, Src_Width,Src_Height,
                           0, Src_Width,Src_Height);
          }
          else {
             ctx.drawImage(image2, 0, height * currentFrame, Src_Width,Src_Height
                           0, Src_Width,Src_Height);
          }
          // change the sprite
          if (currentFrame == frames) {
             currentFrame = 0;
          } else {
             currentFrame++;
          }
      }
      //Moving each step very 40 /msec
      setInterval(draw, 1000/25);
</script>
"""
```

In [10]:

```
HTML(canvas+JSCode)
```

Out[10]:

In [11]:

```
!jupyter nbconvert Canvas.ipynb
```

```
[NbConvertApp] Converting notebook Canvas.ipynb to html
[NbConvertApp] Writing 315284 bytes to Canvas.html
```

In [ ]:

```

```