1 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

1.1 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.

1.2 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**.

2 Android App with HTML

2.1 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 AndroidManifest.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/assets`**

4. Java Codes

```
...
    ...
import android.webkit.WebView;
    ...
public class WebviewActivity extends Activity {
       private WebView webview;
       @Override
       public void onCreate(Bundle savedInstanceState) {
              ...
           webview = (WebView)this.findViewById(R.id.webvie
w1);
           webview.getSettings().setJavaScriptEnabled(true)
;
           webview.loadUrl("file:///android_asset/index.htm
l");
       }
       ...
}
```

4. Kotlin Codes

Mostly as above, a little changes is that private object acclaim was replaced by "var" acclaim:

```
    ...
import android.webkit.WebView;
    ...
class WebViewActivity: AppCompatActivity() {
       var webview: WebView? = null
       @Override
       public void onCreate(Bundle savedInstanceState) {
              ...
           webview = findViewById<WebView>(R.id.webview01)
           webview.getSettings().setJavaScriptEnabled(true)
;
           webview.loadUrl("file:///android_asset/index.htm
l");
       }
       ...
}
```

5. Run App

Done.

2.2 Self Practice

Try to make a html-based app.

3 Canvas

Canvas is one of new supported feature of HTML5 for creating picture online and is the best choice to develop app with mobile webview.

Whatever an image or an animation we want to create and display requires a space to put it on; canvas, considered as a paper, is the platform for it to display on the browser.

3.1 Usage

1. First, acclaim a *canvas* tag with size, width and height;
2. create some objects, line, circle, for instance;
3. make animation etc.

3.2 Template

```
<html>
<head>
<meta name="viewport" content="user-scalable=no, initial-sc
ale=1, maximum-scale=1,
                                minimum-scale=1, width=devic
e-width, height=device-height" />
</head>
<script type="text/javascript">
// acclaim a canvas in javascript and play on HTML camvas s
pace
window.onload = function {
   var a_canvas = document.getElementById("a_canvas");>
   var context = a_canvas.getContext("2d");
   ...
}
</script>
<body>
<!-- acclaim a canvas space in HTML body -->
<canvas id="a_canvas" style="width:100%; height:100%"></can
vas>
</body>
</html>
```

This should add a new canvas as follows:

3.3 Poly-line

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

```
// start sketech
context.beginPath();
// move to the initial position
context.moveTo(x0,y0)
// add a line from (x0,y0) to (x1,y1)
context.lineTo(x1,y1);
...
// clomplet the sketch
context.closePath();
// use the red pencil
context.strokeStyle="red";
// visualize the result
context.stroke();
```

3.4 Simple Pictures

3.4.1 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>
```

3.5 Result picture

3.5.1 Circle

circle centred at (200,200) with radius 30: $(x, y) = (r\cos t, r\sin t)$

```
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);
}
```

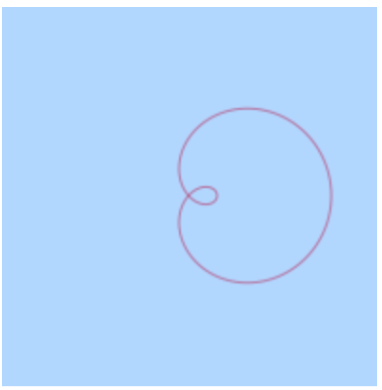
3.5.2 Cardiac

$r = 30 + 45\cos\theta$ and centred at $(x_0, y_0) = (100, 100)$:

$$\mathbf{x = x_0 + r\cos\theta, y = y_0 + r\sin\theta}$$

```
<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>
```

3.6 Result Picture



3.6.1 Self-Practice

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

3.7 Arc For Curves

$$(x - 50)^2 + (y - 50)^2 = 20^2$$

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

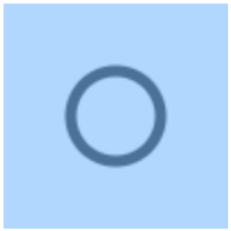
- (x,y): center;
- radius: radius of circle;
- startAngle, endAngle: plot from startAngle to endAngle;
- anticlockwise (true or false): whether plaot in counterclock or not.

```
    context.beginPath();
    context.strokeStyle = "black";
    context.lineWidth = 5;
    context.arc(50, 50, 20, (Math.PI/180)*0, (Math.PI/180)*3
60, false);
    context.stroke();
    context.closePath();
```

3.8 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, false);
         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>
```

3.9 Result Picture



3.10 Canvas Animation

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

3.10.1 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.

3.10.2 Animation Structure

Briefly,

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

Complete Code

```
    <body>
        <canvas id="myCanvas" width="578" height="200"></can
vas>
        <script>
            function init() {
                ### Basic setting for size of picture, line
width, doing animation etc.
            }
            // unify the function of reqAnimFrame for differe
nt browsers and do animation
            function animate() {
                reqAnimFrame = window.requestAnimationFrame ||
                    window.webkitRequestAnimationFrame ||
                    window.mozRequestAnimationFrame    ||
                    window.msRequestAnimationFrame     ||
                    window.oRequestAnimationFrame;
                reqAnimFrame(animate);
                draw();
            }

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

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

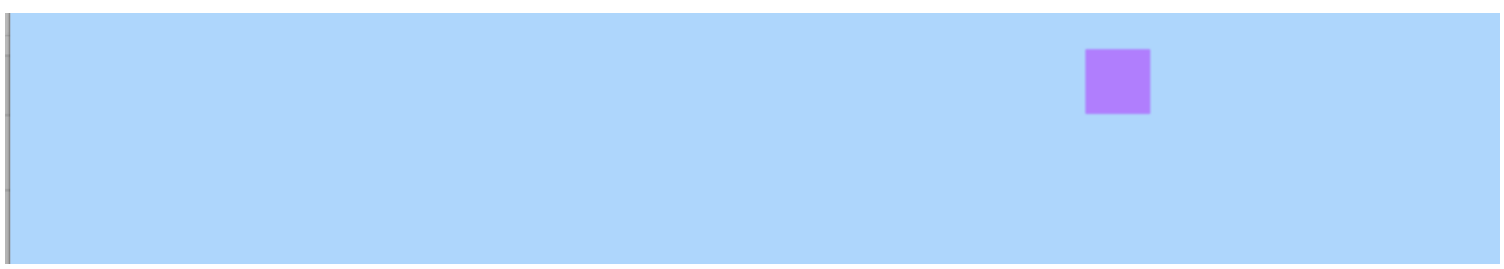
3.11 Note

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.webkitRequestAnimationFrame ||
      window.mozRequestAnimationFrame ||
      window.msRequestAnimationFrame  ||
      window.oRequestAnimationFrame;
    reqAnimFrame( animate );
    draw();
}
```

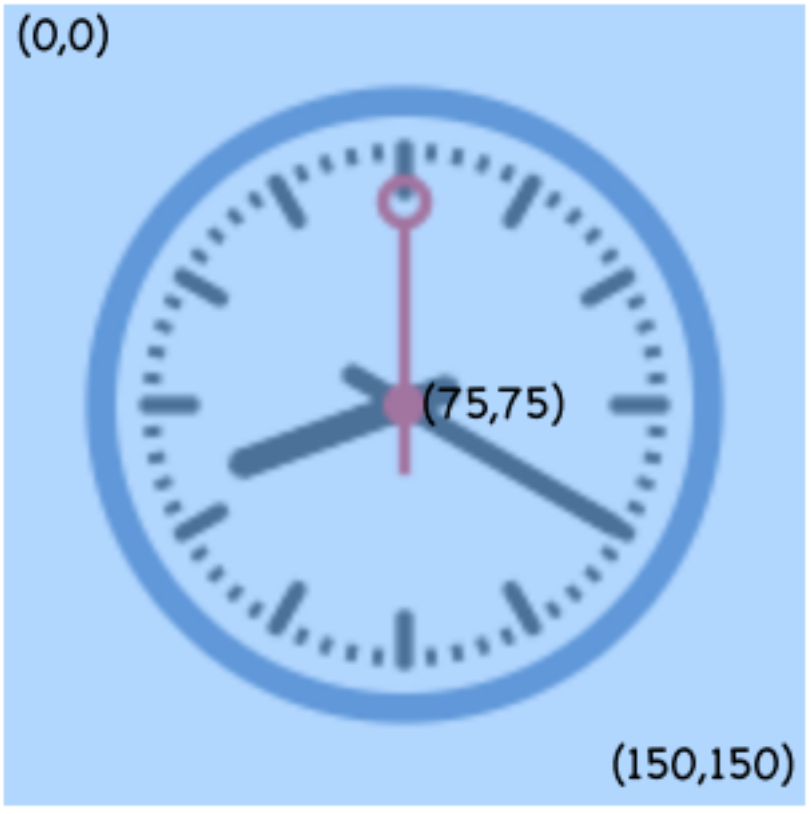
3.12 Moving within Interval

```
  <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.webkitRequestAnimationFrame ||
           window.mozRequestAnimationFrame    ||
           window.msRequestAnimationFrame     ||
           window.oRequestAnimationFrame;
        reqAnimFrame(animate);

        draw();
      }

      function draw() {
        // increase x'velocity by speed (in pixels)
        x += speed;
        // reflect while it hits boundary
        if(x <= 0 || x >= canvas.width-25){
          speed = -speed;
        }
        // refresh the canvas and redraw a purple 25x25 squ
are
        context.clearRect(0, 0, canvas.width, canvas.height
);
        context.fillStyle = "#ff00ff";
        context.fillRect(x, y, 25, 25);
      }
      init();
      animate();
    </script>
```



3.13 Clock

Take a look at the snapshoot



3.14 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)*se
c )
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.webkitRequestAnimationFrame ||
        window.mozRequestAnimationFrame    ||
        window.msRequestAnimationFrame     ||
        window.oRequestAnimationFrame;
        reqAnimFrame( animate );
        clock();
    }
```

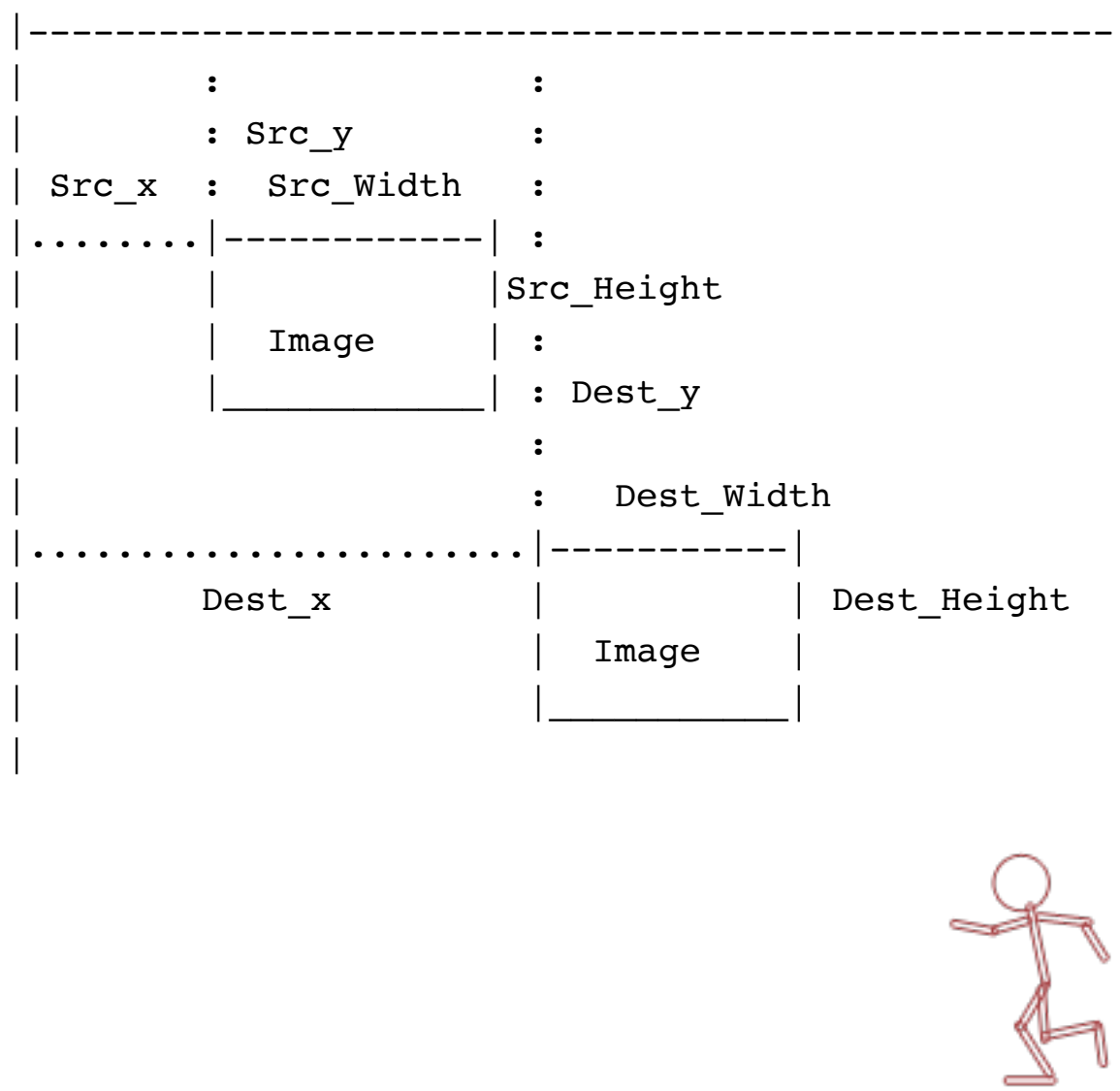
3.15 SpriteAnimation

Another traditional animation, sprite animation: just play the frame1 in sprite, refresh display, play frame2, refresh, and so on.

Additionally, sprite was displayed one by one but move toward the right, turn around while hit the right wall, and so on; it is called a spite animation.

3.16 rendering api

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




```
[NbConvertApp] Converting notebook Canvas.ipynb to htm
l
[NbConvertApp] Writing 345362 bytes to Canvas.html
```