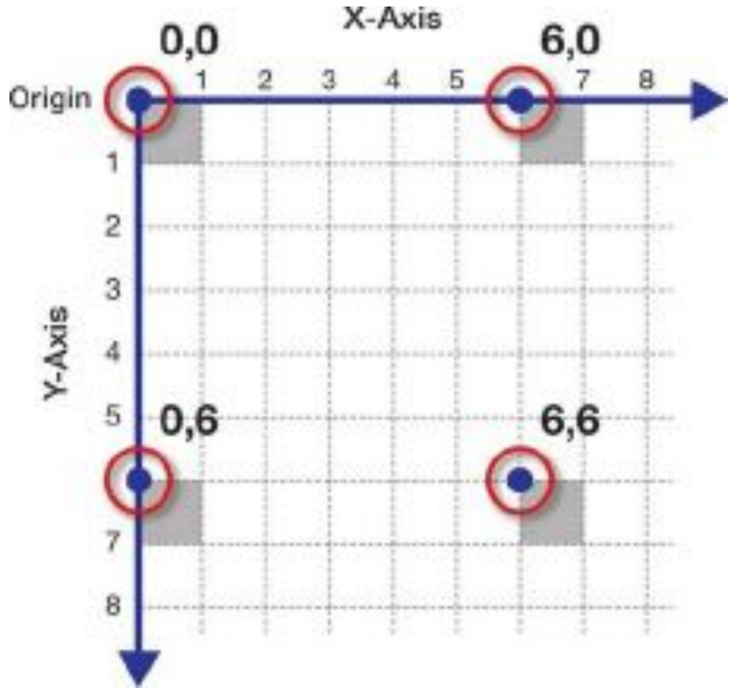
Canvas

CANVAS 介紹

- 畫布, 可讓程式在上面動態產生圖形, 圖表, 圖像和動畫等
- 語法:<canvas></canvas>
- 座標:



Canvas的基本操作

Canvas的寬與高不能在CSS中設定

- Canvas的設定

  ```
  <canvas id="myCanvas" height="800" width="1000" style="border: 3px solid"></canvas>
  ```

- Javascript中取得myCanvas物件

  ```
  const canvas = $("#myCanvas")[0];
  ```

jquery傳回來的是陣列

- 取得myCanvas物件中的渲染環境

  ```
  const ctx = canvas.getContext("2d");
  ```

- 設定渲染環境的新路徑

  ```
  ctx.beginPath();
  ```

- 設定繪圖起始點

  ```
  ctx.moveTo(x, y);
  ```

- 繪出圖形邊框

  ```
  ctx.stroke();
  ```

線, 圓/弧, 方框, 文字

- 線

  ctx.lineTo(x2,y2);
- 圓

  ctx.arc(95,50,40,0,2*Math.PI,false); // arc(x,y,r,start,stop,順/逆時鐘)
- 方框
  - 實體方框

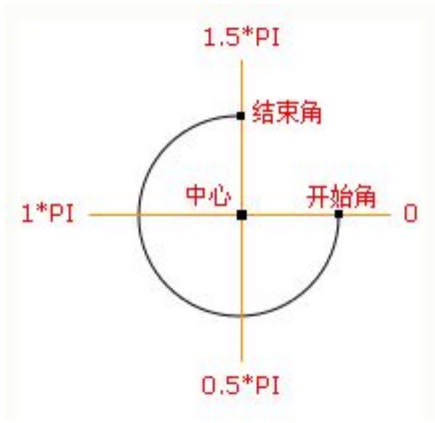
  ctx.fillrect(10, 20, 30, 20); // fillrect(x, y, width, height)
  - 空心方框

  ctx.rect(10, 20, 30, 20); // rect(x, y, width, height)
- 文字
  - 實體文字

  ctx.fillText("Hello World",10,50);
  - 空心文字

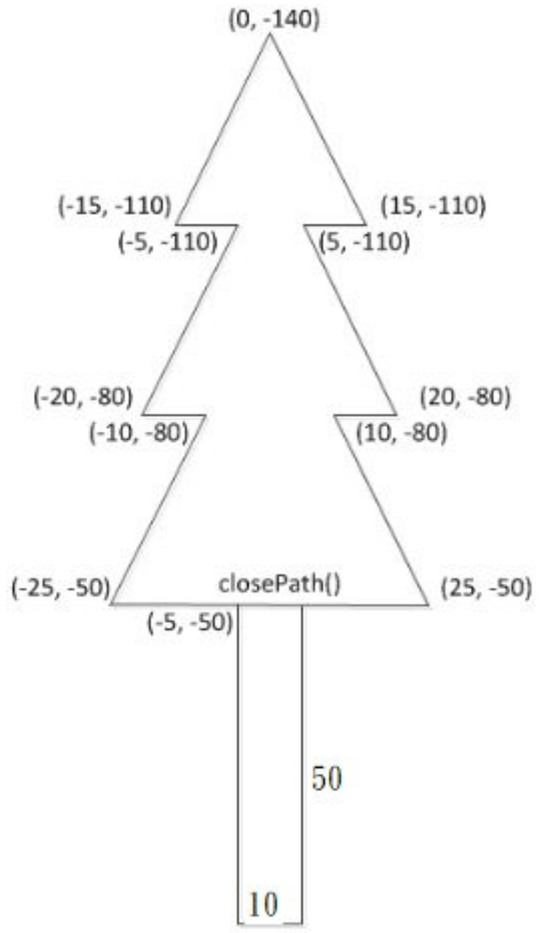
  ctx.strokeText("Hello World",10,50);
  - 字型與大小

  ctx.font = "30px Arial";

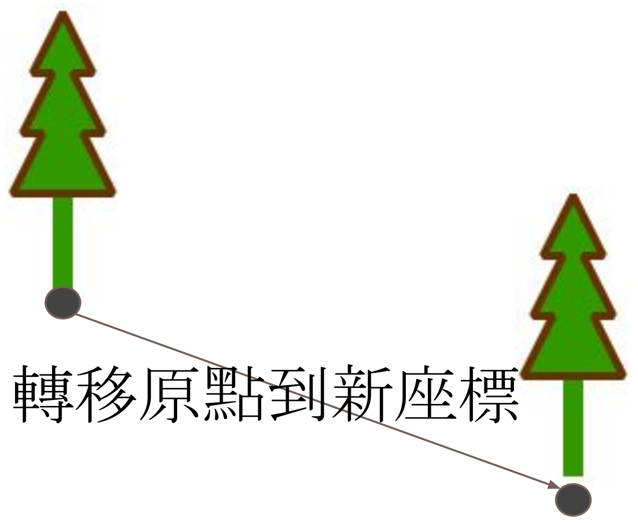


練習—樹

```
ctx.translate(130, 250);
ctx.beginPath();
ctx.moveTo(-25, -50);
ctx.lineTo(-10, -80);
ctx.lineTo(-20, -80);
ctx.lineTo(-5, -110;
ctx.lineTo(-15, -110);
ctx.lineTo(0, -140);
ctx.lineTo(15, -110);
ctx.lineTo(5, -110);
ctx.lineTo(20, -80);
ctx.lineTo(10, -80);
ctx.lineTo(25, -50);
ctx.closePath();
ctx.rect(-5,-50,10, 50);
ctx.stroke();
```

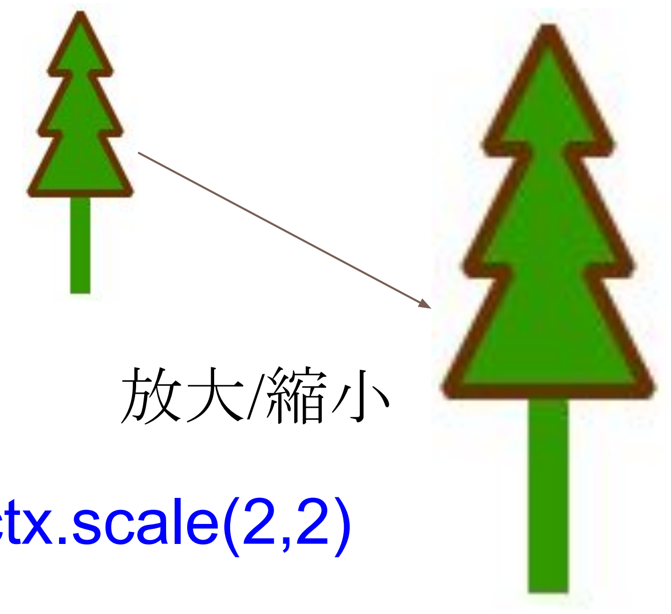


TRANSFORMATION (TRANSLATE, SCALE, ROTATE)



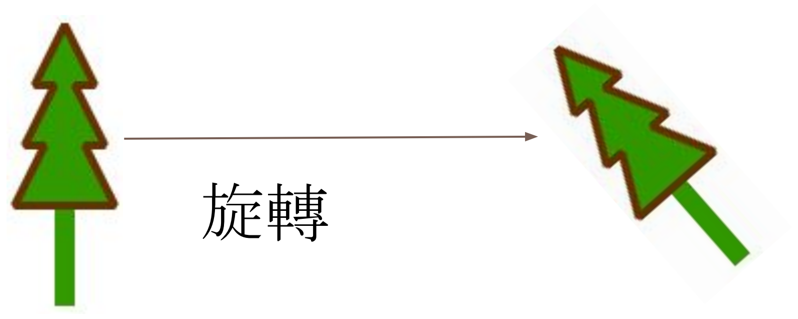
轉移原點到新座標

ctx.translate(100,100)



放大/縮小

ctx.scale(2,2)



旋轉

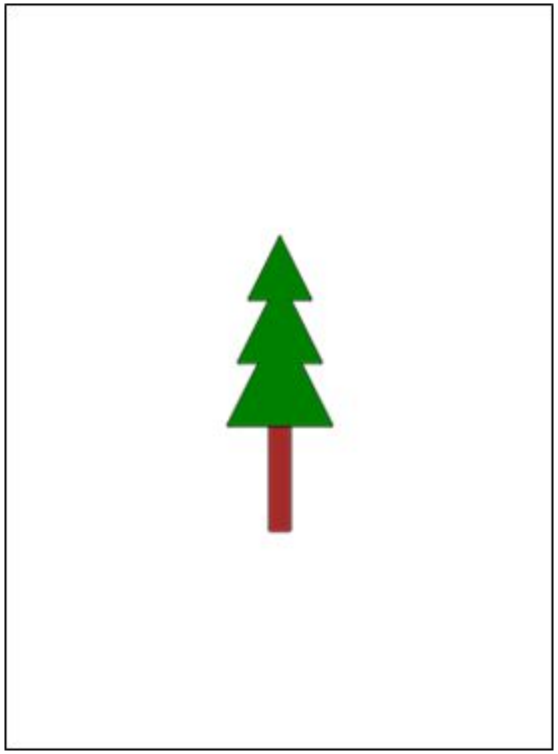
ctx.rotate(20*Math.PI/180);

STROKE STYLE

- Line width
  - ctx.lineWidth = 4;
- Corner style at path joins (round:圓角, bevel:斜角, miter: 斜切)
  - ctx.lineJoin = 'round';
- Line style at endpoints (round, square, butt:預設值)
  - ctx.lineCap = ‘square';
- Stroke style
  - Change color: ctx.strokeStyle = '#663300';
  - Background pattern
- Fill Style
  - Change color: ctx.fillStyle = '#339900';
  - Background pattern
- Fill the region inside all the closed paths
  - ctx.fill();
- Fill rectangular content
  - ctx.fillRect(x, y, w, h); //ex: context.fillRect(-5, -50, 10, 50);

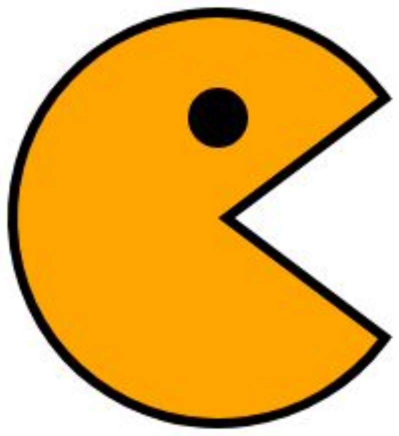
練習─樹 (著色)

```
ctx.translate(130, 250);
ctx.beginPath();
ctx.moveTo(-25, -50);
ctx.lineTo(-10, -80);
ctx.lineTo(-20, -80);
ctx.lineTo(-5, -110;
ctx.lineTo(-15, -110);
ctx.lineTo(0, -140);
ctx.lineTo(15, -110);
ctx.lineTo(5, -110);
ctx.lineTo(20, -80);
ctx.lineTo(10, -80);
ctx.lineTo(25, -50);
ctx.closePath();
ctx.strokeStyle = "black";
ctx.fillStyle = "green";
ctx.stroke();
ctx.fill();
ctx.fillStyle = "brown";
ctx.fillRect(-5,-50,10, 50);
```

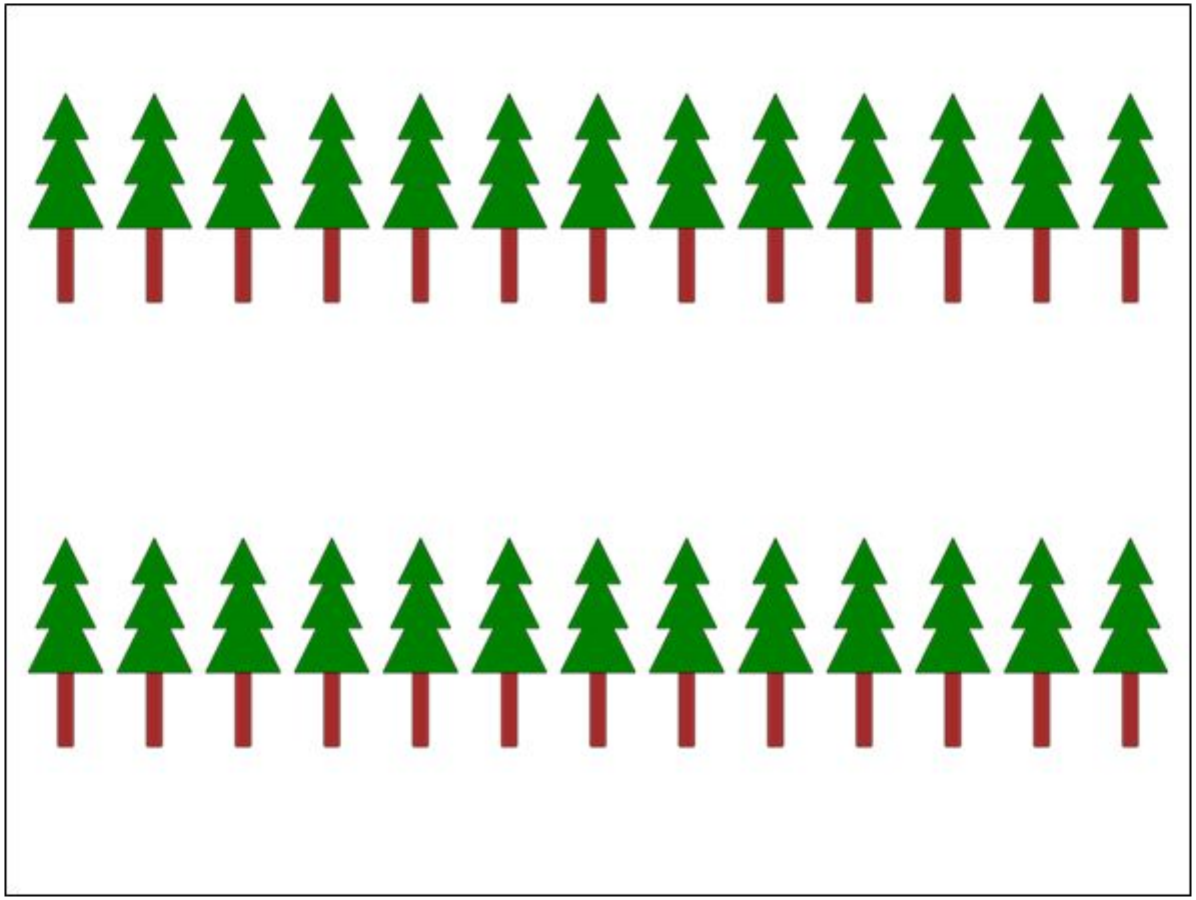


練習─PacMan (著色)

```
ctx.translate(300, 300);
const radian = Math.PI / 180;
ctx.beginPath();
ctx.strokeStyle = 'black';
ctx.fillStyle = 'orange';
ctx.lineWidth = 10;
ctx.moveTo(250, 250);
ctx.arc(250, 250, 100, 37 * radian, 323 * radian, false);
ctx.closePath();
ctx.fill();
// 眼睛
ctx.beginPath();
ctx.fillStyle = 'black';
ctx.arc(250, 200, 10, 0 * radian, 360 * radian, false);
ctx.stroke();
ctx.fill();
```



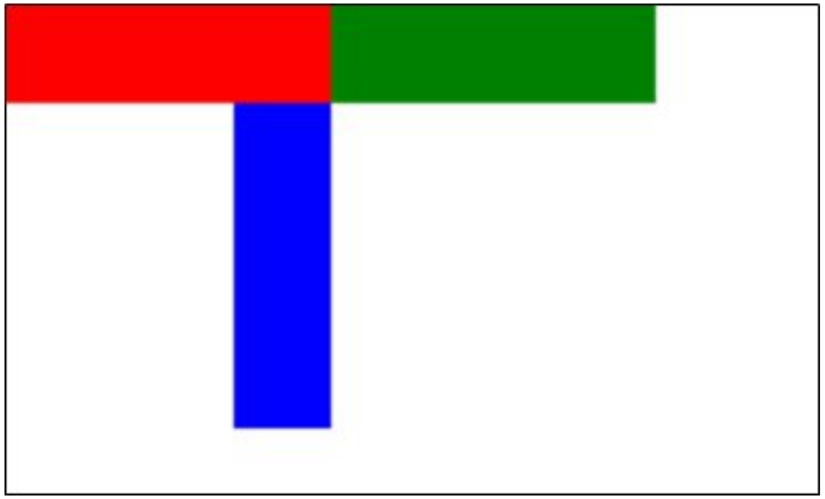
練習——排樹



Excerise

儲存/還原狀態

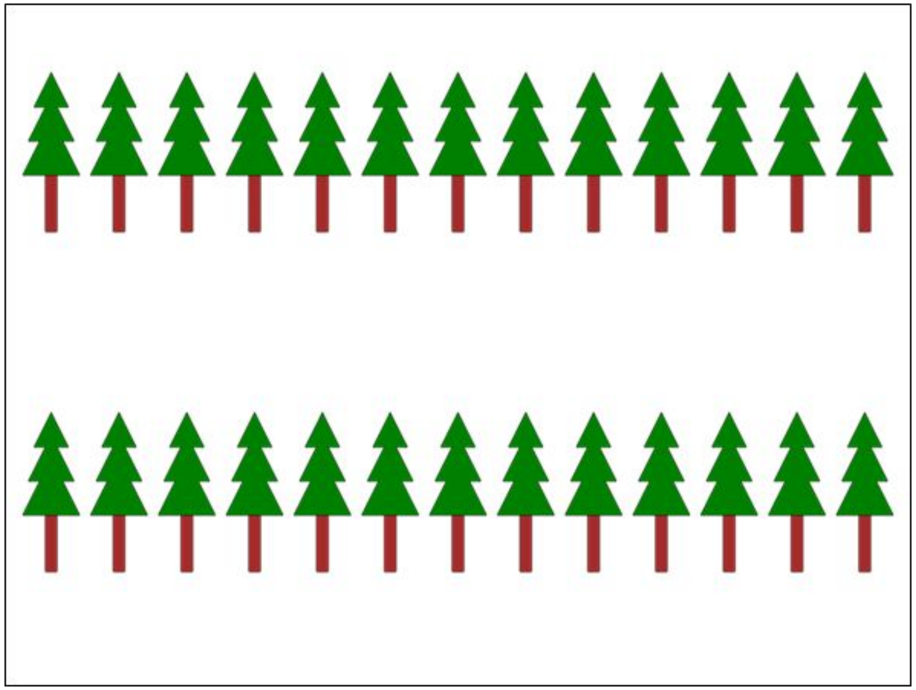
- 儲存狀態
  - ctx.save();
- 還原狀態
  - ctx.restore();



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

 var context= canvas.getContext("2d");
// Draw red rect
context.fillStyle = "red";

 context.fillRect(0, 0, 100, 30);


// Draw blue rect
context.save();
context.translate(100,30);
context.rotate(90*Math.PI / 180);
context.fillStyle = "blue";

 context.fillRect(0, 0, 100, 30);
context.restore();


// Draw green rect
context.translate(100,0);
context.fillStyle = "green";
context.fillRect(0, 0, 100, 30);
```

ANSWER - Draw Street Trees

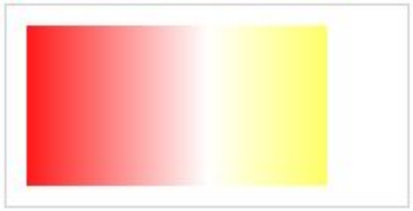
```
function DrawStreetTrees() {
        for(i=40;i<800; i+=60) {
        DrawTree(i, 200);
        DrawTree(i, 500);
}

function DrawTree(x, y) {
        // Save the current drawing state
        context.save();
        // Set start coordinate
        context.translate(x, y);
        // Draw tree trunk
        ....
        // Draw tree leaf
        ....
        // Restore the old drawing state
        context.restore();
}
```

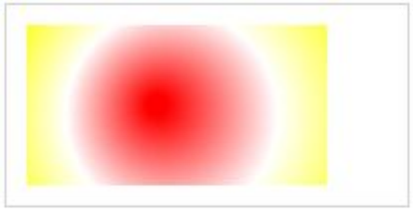


- Linear/Circular Gradient

■ Draw Linear/Circular Gradient

```
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
// Create gradient
// createLinearGradient(x,y,x1,y1) - Linear gradient
var grd = context.createLinearGradient(0,0,200,0);
// createRadialGradient(x,y,r,x1,y1,r1) - Circular gradient
var grd = context.createRadialGradient(75,50,5,90,60,100);
grd.addColorStop(0,"red");
grd.addColorStop(0.5,"white");
grd.addColorStop(1,"yellow");
// Fill with gradient
context.fillStyle = grd;
context.fillRect(10,10,150,80);
```



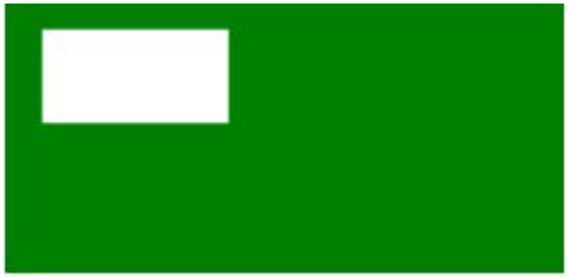


Exercise

PS: addColorStop(): Specify the color stops, and its position along the gradient. Gradient positions can be anywhere between 0 to 1.

CANVAS – clearRect() Method

- The clearRect() method clears the specified pixels within a given rectangle.
  - context.clearRect(x, y, width, height);
  - ex:
    - context.fillStyle="green";
    - context.fillRect(0,0,300,150);
    - context.clearRect(20,20,100,50);



- Clear Canvas
  - context.clearRect(0, 0, canvas.width, canvas.height);

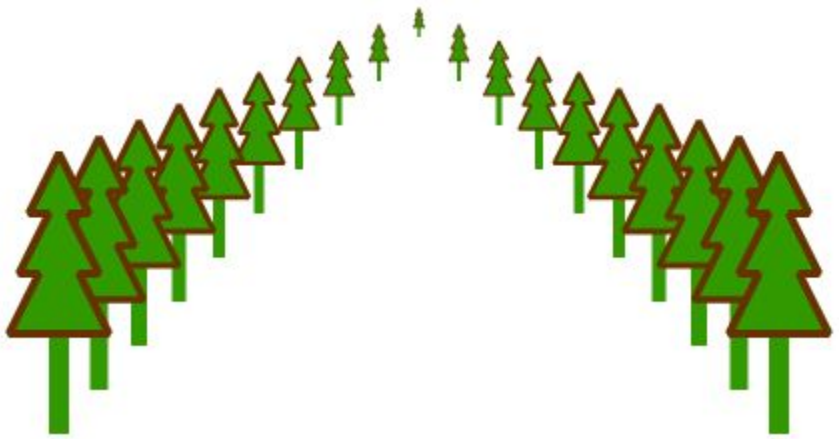
Excerise

- Save the current drawing state

  context.save();

- Restore the old drawing state

  context.restore();



Excerise

```
var canvas;
var context;
function drawStreetTree(){
    canvas = document.getElementById('canvas');
    context = canvas.getContext('2d');
    var m =1.1;
    var k = -400;

    for (var i = 0; i < 10; i++) {
      x_right = i*20 + 500;
      y = m*x_right + k;
      x_left = -i*20 + 500;
      context.save();
      context.translate(x_right, y);
      context.scale(i*0.1+0.1,i*0.1+0.1);
      drawTree();
      context.restore();
      context.save();
      context.translate(x_left, y);
      context.scale(i*0.1+0.1,i*0.1+0.1);
      drawTree();
      context.restore();
    }
}
```

Timer – setInterval() / setTimeout()

- setInterval(*function*, *milliseconds*)
  - The setInterval() method calls a function or evaluates an expression at specified intervals (in milliseconds).
  - The setInterval() method will continue calling the function until clearInterval() is called, or the window is closed.
  - *example*
    - *var timer = setInterval(function(){ alert("Hello"); }, 3000);*
    - *clearInterval(timer);*
- setTimeout(*function*, *milliseconds*)
  - The setTimeout() method calls a function or evaluates an expression after a specified number of milliseconds.
  - *example*
    - *setTimeout(function(){ alert("Hello"); }, 3000);*

Excerise

requestAnimationFrame()

Excerise

Draw Clock

Excerise

ANSWER – Draw Clock

```
// Step 1: Get Canvas  & Making Timer
var canvas, context, timer;
function StartDrawClock() {
        canvas = document.getElementById("canvas");
        context= canvas.getContext("2d");
        timer = setInterval(DrawClock, 1000);
}

// Step 3: Initialize Time Infomation
var sec, min, hour;
function InitTimeInfo() {
        now = new Date();
        sec = now.getSeconds();
        min = now.getMinutes();
        hour = now.getHours();
        hour = hour  > 12 ? hour - 12 : 0 ;
}
```

```
// Step 2: Initialize Canvas & Drow Clock
var radius;
function DrawClock() {
        // Initialize Time Infomation
        InitTimeInfo();
        radius = Math.min(canvas.width/2,canvas.height/2);
        context.save();
        context.clearRect(0,0,canvas.width,canvas.height);
        context.translate(canvas.width/2,canvas.height/2);
        context.scale(0.9,0.9);
        // Rotate -90 Degree
        context.rotate(-Math.PI/2);
        DrawCircle();
        DrawMinLine();
        DrawHourLine();
        DrawHourHand();
        DrawMinuteHand();
        DrawSecondHand();
        context.restore();
}
```

ANSWER – Draw Clock

```
// Step 4: Draw Circle
function DrawCircle() {
        context.save();
        context.strokeStyle="DarkRed";
        context.fillStyle="DarkRed";
        context.lineWidth=5;
        context.beginPath();
        // 1 PI = ½ Circle = 180 Degree
        context.arc(0, 0, radius, 0, 2*Math.PI);
        context.stroke();
        context.restore();
}
```

```
// Step 5: Draw Minute Line
function DrawMinuteLine() {
        context.save();
        context.strokeStyle="gray";
         context.fillStyle="gray";
        context.lineWidth=2;
        context.lineCap="round";
        context.beginPath();
        for(var i=0;i<60;i++){
                context.rotate(Math.PI/30);
                context.moveTo(radius-20,0);
                context.lineTo(radius-10,0);
        }
        context.stroke();
        context.restore();
}
```

ANSWER – Draw Clock

```
// Step 6: Draw Hour Line
function DrawHourLine() {
        context.save();
        context.strokeStyle="black";
        context.fillStyle="black";
        context.lineWidth=3;
        context.lineCap="round";
        context.beginPath();
        for(var i=0;i<12;i++){
                context.rotate(Math.PI/6);
                context.moveTo(radius-30,0);
                context.lineTo(radius-10,0);
        }
        context.stroke();
        context.restore();
}
```

```
// Step 7: Draw Hour Hand
function DrawHourHand() {
        context.save();
        context.strokeStyle="Navy";
        context.fillStyle="Navy";
        context.lineWidth=4;
        context.lineCap="butt";
        context.beginPath();
        context.rotate(hour*(Math.PI/6)
        +min*(Math.PI/360) + sec*(Math.PI/21600));
        context.moveTo(-10,0);
        context.lineTo(radius*0.5,0);
        context.stroke();
        context.restore();
}
```

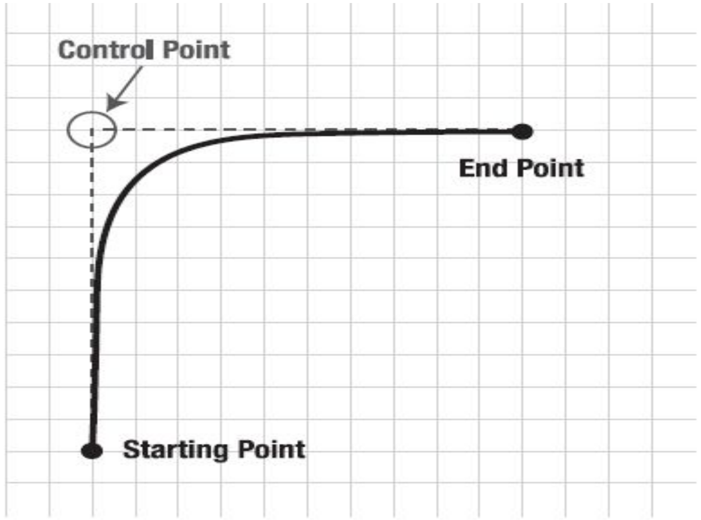
ANSWER – Draw Clock

```
// Step 8: Draw Minute Hand
function DrawMinuteHand() {
        context.save();
        context.strokeStyle="DodgerBlue";
        context.fillStyle="DodgerBlue";
        context.lineWidth=4;
        context.lineCap="butt";
        context.beginPath();
        context.rotate(min*(Math.PI/30) +
        sec*(Math.PI/1800));
        context.moveTo(-20,0);
        context.lineTo(radius*0.7,0);
        context.stroke();
        context.restore();
}
```

```
// Step 9: Draw Second Hand
function DrawSecondHand() {
        context.save();
        context.strokeStyle="red";
        context.fillStyle="red";
        context.lineWidth=2;
        context.lineCap="butt";
        context.beginPath();
        context.rotate(sec*(Math.PI/30));
        context.moveTo(-30,0);
        context.lineTo(radius*0.9,0);
        context.stroke();
        context.restore();
}
```

QUADRATIC CURVE

- Starting Point: current location
- context.quadraticCurveTo(ControlPointX, ControlPointY, EndPointX, EndPointY);
- Example:
  - context.save();
  - context.translate(-10, 350);
  - cucontext.moveTo(0, 0);   //Start point
  - //Control point: (170,-50) End Point: (260, -190)
  - context.quadraticCurveTo(170, -50, 260, -190);
  - //Control point: (310,-250) End Point: (410, -250)
  - context.quadraticCurveTo(310, -250, 410, -250);
  - context.lineWidth = 20;
  - context.strokeStyle = '#663300';
  - ontext.stroke();
  - context.restore();



Excerise

- Usage
  - context.createPattern(image, repeat)
    - repeat - repeat, repeat-x, repeat-y, no-repeat
- Example

```
var gravel = new Image();
gravel.src = "gravel.jpg";
context.save();
context.translate(-10, 390);
gravel.onload = function(){
  context.beginPath();
  context.moveTo(0, 0);
  context.quadraticCurveTo(170, -50, 260, -190);
  context.quadraticCurveTo(310, -250, 410, -250);
  context.lineWidth = 20;
  context.strokeStyle = context.createPattern(gravel, 'repeat');
  context.stroke();
  context.restore();
}
```

Excerise

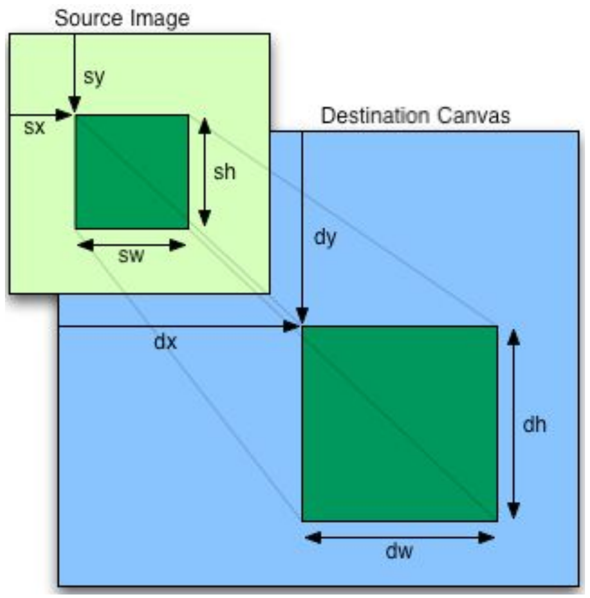
- Load image

  ```
  var img = new Image();
  img.src = "bark.jpg";
  ```
- Confirm the image is loaded

  ```
  img.onload = function(){
    //Draw image onto canvas
  }
  ```
- Draw image onto canvas

  ```
  context.drawImage(image, dx, dy)
  context.drawImage(image, dx, dy, dw, dh)
  context.drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh)
  ```
- Example

  ```
  var bark = new Image();
  bark.src = "bark.jpg";
  bark.onload = function(){
      context.drawImage(bark, -5, -50, 10, 50);
      context.stroke();
      context.restore();
  }
  ```

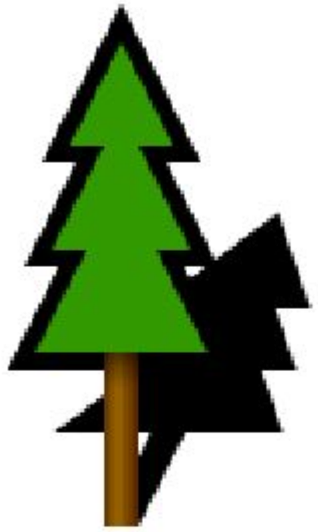


Excerise

- Context.transform(rx, sy, sx, ry, dx, dy)
  - rx – width scale ratio
  - ry – height scale ratio
  - sy – vertical shear
  - sx – horizontal shear

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & S_x & 0 \\ S_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Example
  - var canvas = document.getElementById('canvas');
  - var context = canvas.getContext('2d');
  - context.save();
  - context.translate(130, 250);
  - context.transform(1, 0, -0.5, 1, 0, 0);
  - context.scale(1, 0.6);
  - context.fillStyle = 'rgb(0, 0, 0, 0.2)';
  - context.fillRect(-5, -50, 10, 50);
  - createCanopyPath(context);
  - context.fill();
  - context.restore();

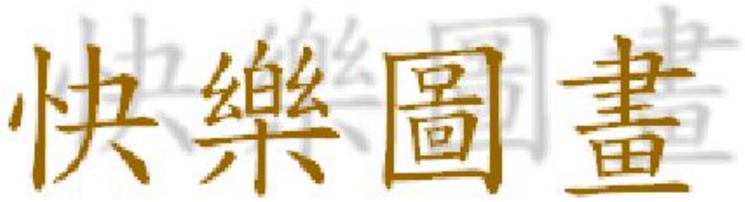


Excerise

- Usage
  - context.fillText(text, x, y, maxwidth)
  - context.strokeText(text, x, y, maxwidth)
  - Property
    - context.font = Font String
    - context.textAlign = start, end, left, right, center
    - context.textBaseLine = top, middle, bottom, …
- Example
  - context.save();
  - context.font = '60px 標楷體';
  - context.fillStyle = '#996600';
  - context.textAlign = 'center';
  - context.fillText('快樂圖畫', 200, 60, 400);
  - context.restore();

Excerise

- Usage
  - shadowColor – Any CSS Color
  - shadowOffsetX – Pixel Count
  - shadowOffsetY – Pixel Count
  - Shadowblur – Gaussian blur
- Example
  - context.shadowColor = 'rgba(0, 0, 0, 0.2)';
  - context.shadowOffsetX = 15;
  - context.shadowOffsetY = -10;
  - context.shadowBlur = 2;
  - context.font = '60px 標楷體';
  - context.fillStyle = '#996600';
  - context.textAlign = 'center';
  - context.fillText('快樂圖畫', 200, 60, 400);



Excerise