v = v0 + a*t

s = v * t + 1/2 *a * t^2

```
只要是设置了状态以后，再接下来的绘制，都是基于我们设置的好的状态
绘制状态：strokeStyle / fillStyle / lineWidth / 变换
```

- 人眼识别频率为60hz（一秒刷新60次），但手机捕捉频率高，所以拍摄电视或电脑屏幕时，会有很多线，
- js代码执行极快，但浏览器去渲染js操作较慢
- DOM的重绘、重排
- setInterval：不同浏览器限定的最小时间不同，有的4ms，有的10ms

```
// 语法：
var lastFrameTime = new Date();
var fn = function() {
    var curFrameTime = new Date();
    console.log( curFrameTime - lastFrameTime );
    lastFrameTime = curFrameTime;
   // 在函数的最后面，调用 requestAnimationFrame 函数
   // 函数接受一个函数参数，就是告诉浏览器下一次刷新的时候，
   // 调用函数 fn !
   window.requestAnimationFrame( fn );
};
fn();
```

---

requestAnimationFrame（）：浏览器只要刷新，就会执行 所有 requestAnimationFrame（）里指定的回调函数；

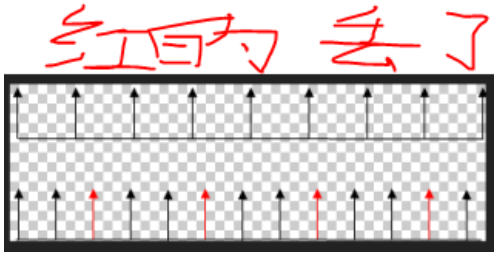
```
// window.requestAnimationFrame()
//
// 这个方法是，HTML5中专门为js实现动画效果提供的一个函数
// 也就是推荐使用 requestAnimationFrame 来实现js中的动画效果

// setInterval的劣势：
// 1 定时器指定的时间是不准确的
// 2 执行动画效果会造成性能的问题，如果页面中使用了多个定时器
//   每个定时器中都来操作DOM元素，操作DOM频繁了以后，会造成严重性能问题
//   DOM的重绘和重排（影响web程序的两个重要因素）
// 3 会造成丢帧的问题
//   刷新频率：  60HZ（一秒钟刷新60次）  ==> 16.67ms
        60Fps
/*setInterval(function() {

}, 1);
// 4ms / 10ms*/
```

想要不丢帧，必须16.67ms内做一帧

```
requestAnimationFrame 的作用：
能够每隔一段时间来调用一个回调函数执行某段代码逻辑

优势：
1 requestAnimationFrame 是根据当前浏览器的刷新频率来进行调用的
  也就是说：浏览器刷新一次，就会将所有的 requestAnimationFrame
  中指定的回调函数执行一次！
  那么，就不会造成丢帧问题了

2 页面中所有的 requestAnimationFrame 会在浏览器刷新的时候，一次性
  全部执行

3 因为 requestAnimationFrame 方法会根据当前浏览器的刷新频率来
  执行，所以，时间是不固定的，但是，不会有问题
```

```
// 语法:
// var lastFrameTime = new Date();
var fn = function() {
  // var curFrameTime = new Date();
  // console.log( curFrameTime - lastFrameTime );
  // lastFrameTime = curFrameTime;

  // 在函数的最后面，调用 requestAnimationFrame 函数
  // 函数接受一个函数参数，就是告诉浏览器下一次刷新的时候，
  // 调用函数 fn !
  window.requestAnimationFrame( fn );
};

fn();
```

丢帧：上面是浏览器的刷新频率，每次刷新只能捕捉到，最后改变的一帧来实现动画，如果1次刷新里，存在多帧，只能捕捉最后一个，如下图，红色的都丢掉了



---

requestAnimationFrame（）修改兔女郎案例：（改完之后贼快，因为1秒刷新60次）

```
var render = function() {

  ctx.clearRect(0, 0, cv.width, cv.height)
  ctx.drawImage(img, frameIndex++ * imgW, 0, imgW, imgH, 100, 100, imgW
    *2, imgH);
  frameIndex %= 4;

  window.requestAnimationFrame( render );
};
render();
```

控制requestAnimationFrame（）的时间

```
var raf = function(callback, interval) {
  // 思路:
  // 获取当前时间与上一帧时间的差值，如果这个插值达到了
  // interval，此时，就调用 callback 函数，执行 callback 中的代码逻辑

  // 12:00 0000
  var lastFrameTime = new Date();
  (function render() {
    // 12:00 100
    // 12:00 200
    // 12:00 300
    var curFrameTime = new Date();
    if( curFrameTime - lastFrameTime >= interval ) {
      // 12:00 200，将当前时间赋值给上一帧时间
      // 这样 下一帧 将去 上一帧 才能准确获取到代码执行的时间间隔
      lastFrameTime = curFrameTime;

      callback();
    }

    window.requestAnimationFrame( render );
  })();

};

// 调用函数
```

```
// 调用函数
raf(function() {
  console.log(123)
}, 1000);
```

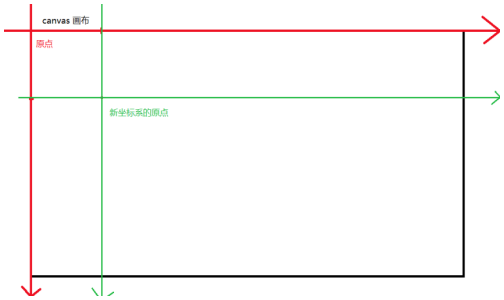
---

canvas中的变换：重要，都是属性状态，类似strokeStyle等状态，不是路径

这三种变换都是对canvas的坐标系做变换，坐标系就变了

- 平移：ctx.translate（x，y）
- 旋转：ctx.rotate（弧度）
- 缩放：ctx.scale（1,1）

1：ctx.translate(100,100)：平移变换后，再绘制内容，就按照新坐标系的坐标进行绘制



2：旋转后，坐标轴也是跑偏的

```
// 2 旋转变换，按照当前坐标系的原点进行旋转
ctx.rotate( toRadian(30) );
ctx.fillRect(100, 0, 100, 100)
```

3：缩放，缩放后坐标变，所以距离和宽高都会变，不是自己想象的缩放图形

scale（x，y）

```
ctx.scale(2, 2);
ctx.fillStyle = 'red'
ctx.fillRect(100, 100, 100, 100)
```

---

canvas是基于状态绘图的：

```
// canvas 绘图是基于状态绘图的
// 也就是，我们设置好一个状态以后，那么，接下来绘制的内容，都是
// 在这个状态的基础上，来绘制的！！！

// 设置状态：
ctx.strokeStyle = 'red';

ctx.rect(0, 0, 100, 100);

ctx.moveTo(200, 200)
ctx.lineTo(400, 200)

ctx.stroke();
```

每次需要画其他路径，改变状态，都需要再设置状态

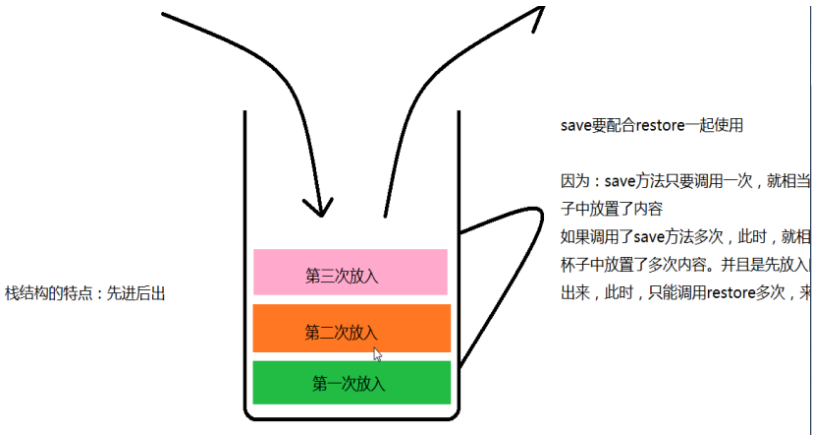
```
// canvas中提供两个方法：
// ctx.save()          表示保存当前的绘制状态
// ctx.restore()       表示恢复到上一次保存的状态
//
// 一般情况下，这两个方法都是配合来使用的

// 保存所有的默认状态
ctx.save();

// 改变状态
ctx.strokeStyle = 'red';
ctx.lineWidth = 10;
ctx.moveTo(100, 100)
ctx.lineTo(200, 100)
ctx.stroke();
```

一般情况都是将默认状态保存起来，然后再通过restore恢复，就不需要重新设置了（好像用在定时器里 或 坐标轴发生变化的情况）

```
// 保存所有的默认状态
ctx.save();

// 改变状态
ctx.strokeStyle = 'red';
ctx.lineWidth = 10;
ctx.moveTo(100, 100)
ctx.lineTo(200, 100)
ctx.stroke();


ctx.beginPath();
// 恢复到上一次保存的状态（默认状态）
ctx.restore();
// ctx.strokeStyle = 'black';
// ctx.lineWidth = 1;
ctx.moveTo(100, 200)
ctx.lineTo(200, 200)
ctx.stroke();
```

save保存的状态是以栈的形式存储，如果调用save两次，需要restore两次才能取出需要的

最佳实践：save一次，restore一次



每次保存的都是清除画布，即使后来画布变了，旋转的是依次增大，第一次0-3度，第二次0-6度，第三次0-9度

```
var startAngle = 0;
var step = 3;

setInterval(function() {
  ctx.clearRect(0, 0, cv.width, cv.height)
  ctx.save();

  ctx.translate(cv.width/2, cv.height/2);
  ctx.rotate( toRadian(startAngle+=step) )
  ctx.strokeRect(-50, -50, 100, 100);

  ctx.restore();
}, 50);
```

```
// 4 刮奖
var flag = false;
// 设置一个开关，用来控制事件中的代码是否执行
cv.addEventListener('mousedown', function() {
  flag = true;
});

cv.addEventListener('mousemove', function() {
  if(flag) {
    // console.log('执行了')

  }
});

cv.addEventListener('mouseup', function() {
  flag = false;
})
```

---

ctx.globalCompositeOperation

```
// 用来控制两个重叠部分以什么样的形式展示，
// destination-out 表示：重叠部分变为透明色
ctx.globalCompositeOperation = 'destination-out';
```

base64编码的 字符串，就是用一个字符串来表示一个图片，浏览器可以将其解析为小图片

```
background-image: url(
  AYAAAAe2bNZAAAAGXRFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAAyRpVFh0WE1M
  OmNvbS5hZG9iZS54bXAAAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVzVNME1wQ2VoaUh
  6cmVTek5UY3prYzlkIj8+IDx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iIHg6eG
```

这就相当于图片了，不需要发请求，得到图片，浏览器可以直接把这段字符串解析为图像

toDataURL:

1.10.4. 画布保存base64编码内容

- 把canvas绘制的内容输出成base64内容。
- 语法：canvas.toDataURL(type, encoderOptions);
- 例如：canvas.toDataURL("image/jpg",1);
- 参数：
  - type，设置输出的类型，比如 image/png image/jpeg等
  - encoderOptions： 0-1之间的数字，用于标识输出图片的质量，1表示无损压缩，类型为： image/jpeg 或者 image/webp 才起作用。

```
var canvas = document.getElementById("canvas");
var dataURL = canvas.toDataURL();
console.log(dataURL);
// "
// blAAAADElEQVQImWNgoBMAAABpAAFEI8ARAAAAAElFTkSuQmCC"

var img = document.querySelector("#img-demo");//拿到图片的dom对象
img.src = canvas.toDataURL("image/png");      //将画布的内容给图片标签显示
```

注意：toDataURL不是canvas的API，需要用对象调用此方法，也就是canvas标签

image/png：固定格式

```
// 2 将奖品信息绘制到画布中
ctx.textAlign = 'center';
ctx.textBaseline = 'middle';
ctx.font = '40px consolas';
ctx.fillStyle = prize.color;
ctx.fillText(prize.name, cv.width/2, cv.height/2);

// 5 将canvas画布转化为图片字符串，然后设置为当前画布的背景
var srcStr = cv.toDataURL('image/png', 1)
cv.style.backgroundImage = 'url(' + srcStr + ')';
```

1表示无损压缩，表示图片质量，取值范围0-1

---

drawImage的其他用法：

```
// drawImage()
// 1 绘制图片
// 2 绘制视频
// 3 绘制其他画布，提高canvas渲染的性能
```

将画布会知道其他画布，提高性能，画布2只画不渲染，都画完，再统一绘制到画布1，画布1渲染

```
// 这个方法还能够绘制 视频 和 另外一个画布

// 绘制视频
window.onload = function() {
  var timerId;

  var v = document.getElementById('v')
  v.addEventListener('play', function() {
    console.log('视频播放 开始 了')
    // ctx.drawImage(v, 0, 0);
    timerId = setInterval(function() {
      ctx.drawImage(v, 0, 0);
    }, 100);
  });

  v.addEventListener('pause', function() {
    console.log('视频播放 暂停 了')
    clearInterval( timerId );
  });
```

```
// drawImage 能够将一个画布绘制到另一个画布中
// 对于有性能要求的功能中，可以通过两个canvas画布配合来实现
//
// 一个画布是隐藏的，我们就在这个隐藏的画布中绘制内容，将所有内容绘制好
// 以后，直接将隐藏画布绘制到 展示给用户看的画布中

var cv1 = document.createElement('canvas')
var ctx1 = cv1.getContext("2d")
ctx1.fillRect(0, 0, 100, 100)
ctx1.fillRect(100, 0, 100, 100)
ctx1.fillRect(0, 100, 100, 100)

// 将隐藏画布一次性绘制到展示的画布中，能够提高渲染性能
// 注意：参数为画布（不是上下文）
ctx.drawImage(cv1, 0, 0);
```

下面都是属性

- lineCap： 线帽，线两边圆角
- lineJoin： 线相交，相交时，角的样子

- miterLimit：夹角长度
- setLineDash（[5，10，3]）：第一个值对应实线的，第二个值虚线，第三个值实现，循环下去
    - lineDashOffset（）
- 设置阴影

```
设置阴影
ctx.fillStyle = 'rgba(255,0,0, .9)'
  ctx.shadowColor = 'teal';
  ctx.shadowBlur = 10; // 模糊级别
  ctx.shadowOffsetX = 10;
  ctx.shadowOffsetY = 10;
  ctx.fillRect(100, 100, 100, 100);
```

- 设置线性和径向渐变

```
var rlg = ctx.createRadialGradient(300,300,10,300,300,200);
rlg.addColorStop(0, 'teal');     //添加一个渐变颜色
// rlg.addColorStop(.4, 'navy');
rlg.addColorStop(1, 'purple');
ctx.fillStyle = rlg;//设置 填充样式为延续渐变的样式
ctx.fillRect(100, 100, 500, 500);
```

- 设置背景

```
// 设置背景
// var img = document.getElementById('lamp');
var img = new Image();
img.src = 'imgs/2.jpg';
img.onload = function() {
  var pat = ctx.createPattern(img, 'repeat-x');
  // ctx.rect(0,0,cv.width/2,cv.height/2);
  ctx.rect(0,0,cv.width,cv.height);
  ctx.fillStyle = pat;//  把背景图设置给填充的样式
  ctx.fill();
  // ctx.clearRect(0, 0, cv.width, cv.height)
};
```

- 限定剪裁区域
- 全局不透明度

重绘重排

转换

刮刮乐

小鸟