

HTML5程序设计基础

第十二章 画布（三）



主要内容

01

渐变色

02

图片填充与合成

03

图片绘制

04

像素操作

05

动画循环

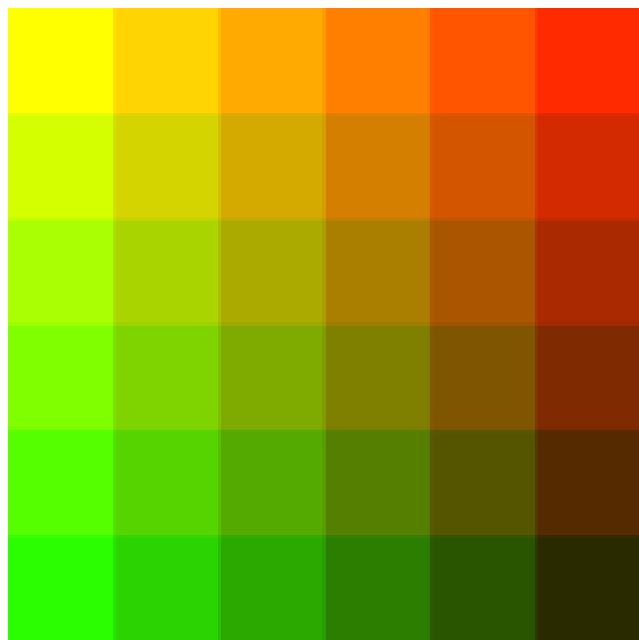
01

渐变色



颜色

- ✦ fillStyle 设置用于填充绘画的颜色。
- ✦ strokeStyle 设置用于绘制描边的颜色。



demo12-1.html

demo12-2.html

线性渐变色

- ✦ canvas 支持线性渐变和放射渐变。
- ✦ **createLinearGradient(x0,y0,x1,y1)** 创建**线性渐变对象**。
使用该对象作为 strokeStyle 或 fillStyle 属性的值。

```
context.createLinearGradient(x0,y0,x1,y1);
```

参数	描述
x0	渐变开始点的 x 坐标
y0	渐变开始点的 y 坐标
x1	渐变结束点的 x 坐标
y1	渐变结束点的 y 坐标

线性渐变色

🎨 **addColorStop**(stop,color)规定渐变对象中的颜色和停止位置。

- 如果不对 gradient 对象使用该方法，渐变将不可见。为了获得可见的渐变，需要创建至少一个色标。
- 可以多次调用此方法来改变渐变。

```
gradient.addColorStop(stop,color);
```

参数	描述
stop	介于 0.0 与 1.0 之间的值，表示渐变中开始与结束之间的位置。
color	在结束位置显示的 CSS 颜色值

线性渐变色

- 两个临近 stop 距离之间的颜色是过渡颜色。
- 小于最小 stop 的部分会按最小 stop 的 color 来渲染，大于最大 stop 的部分会按最大 stop 的 color 来渲染。
- 渐变可用于填充矩形、圆形、线条、文本等等。

放射状/环形渐变

 **createRadialGradient**(x0,y0,r0,x1,y1,r1)

创建放射状/环形的渐变对象。

```
context.createRadialGradient(x0,y0,r0,x1,y1,r1);
```

参数	描述
x0	渐变的开始圆的 x 坐标
y0	渐变的开始圆的 y 坐标
r0	开始圆的半径
x1	渐变的结束圆的 x 坐标
y1	渐变的结束圆的 y 坐标
r1	结束圆的半径

02

图片填充与合成



图案填充、描绘

createPattern() (image, type)方法

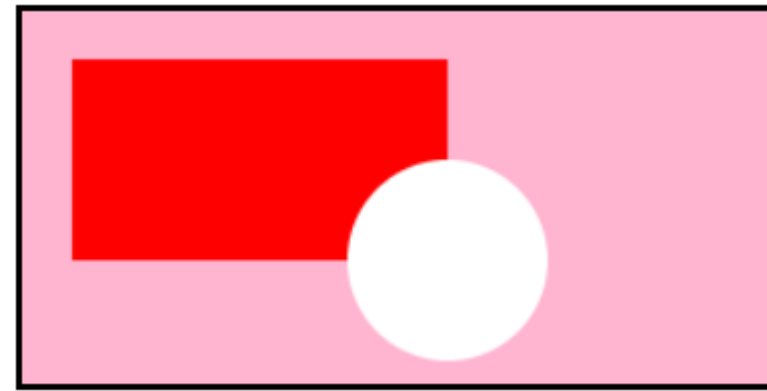
- 在指定的方向内重复指定的元素
- 元素可以是图片、视频，或者其他 <canvas> 元素
- 被重复的元素可用于绘制/填充矩形、圆形或线条等

参数	描述
image	规定要使用的图片、画布或视频元素。
repeat	默认。该模式在水平和垂直方向重复。
repeat-x	该模式只在水平方向重复。
repeat-y	该模式只在垂直方向重复。
no-repeat	该模式只显示一次（不重复）。

合成操作

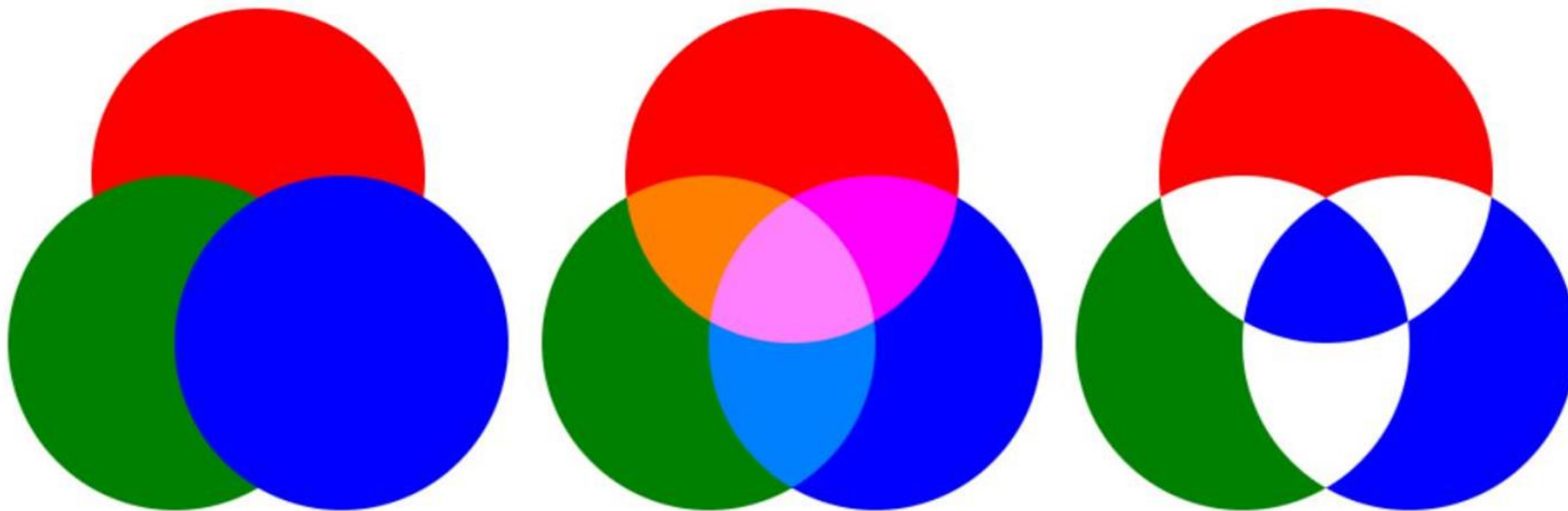
🎨 globalCompositeOperation 属性

- 设置如何将一个源图像绘制到目标图像上。
- 源图像 —— 打算放置到画布上的绘图。
- 目标图像 —— 已经放置在画布上的绘图。



🎨 属性取值：`source-over` | `source-atop` | `source-in` | `source-out` | `destination-over` | `destination-atop` | `destination-in` | `destination-out` | `lighter` | `copy` | `xor`

练习



03

图片绘制



绘制图像

- 在 HTML5 中，不仅可以使使用 Canvas API 来绘制图形，还可以读取磁盘或网络中的图像文件，然后使用 Canvas API 将该图像绘制在画布中。

绘图

 **drawImage()** 方法在画布上绘制图像或视频。

➤ 语法1：在画布上定位图像

```
context.drawImage(img, x, y)
```

➤ 语法2：在画布上定位图像，并规定图像的宽度和高度（缩放）

```
context.drawImage(img, x, y, width, height)
```

参数	描述
x	在画布上放置图像的 x 坐标位置。
y	在画布上放置图像的 y 坐标位置。
width	可选。要使用的图像的宽度。（伸展或缩小图像）
height	可选。要使用的图像的高度。（伸展或缩小图像）

绘图

注意：需要在图片加载完成之后才能绘制图片。

```
var img = new Image(); //新建一个IMG对象
img.src='cake.jpg';
img.onload=function ( ){
    context.drawImage(img,50,100,300,300);
}
```

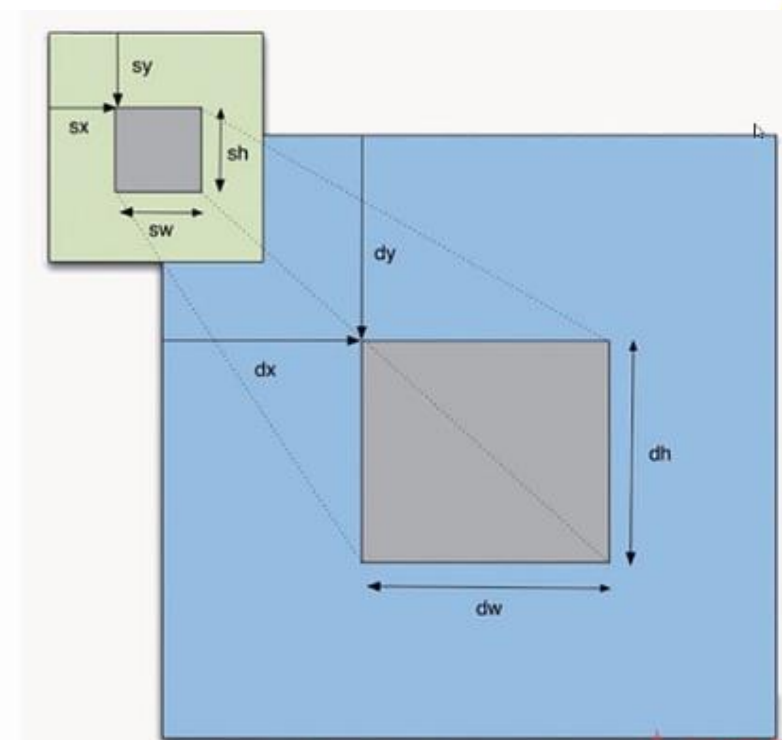

绘图

- 🎨 语法3：剪切图像，并在画布上定位被剪切的部分。（剪切、缩放）

```
context.drawImage(img, sx, sy, sw, sh, dx, dy, dwidth, dheight)
```

参数	描述
img	规定要使用的图像、画布或视频。
sx	可选。开始剪切的 x 坐标位置。
sy	可选。开始剪切的 y 坐标位置。
swidth	可选。被剪切图像的宽度。
sheight	可选。被剪切图像的高度。

drawImage()该方法的参数只能为 3 5 9



demo12-9.html

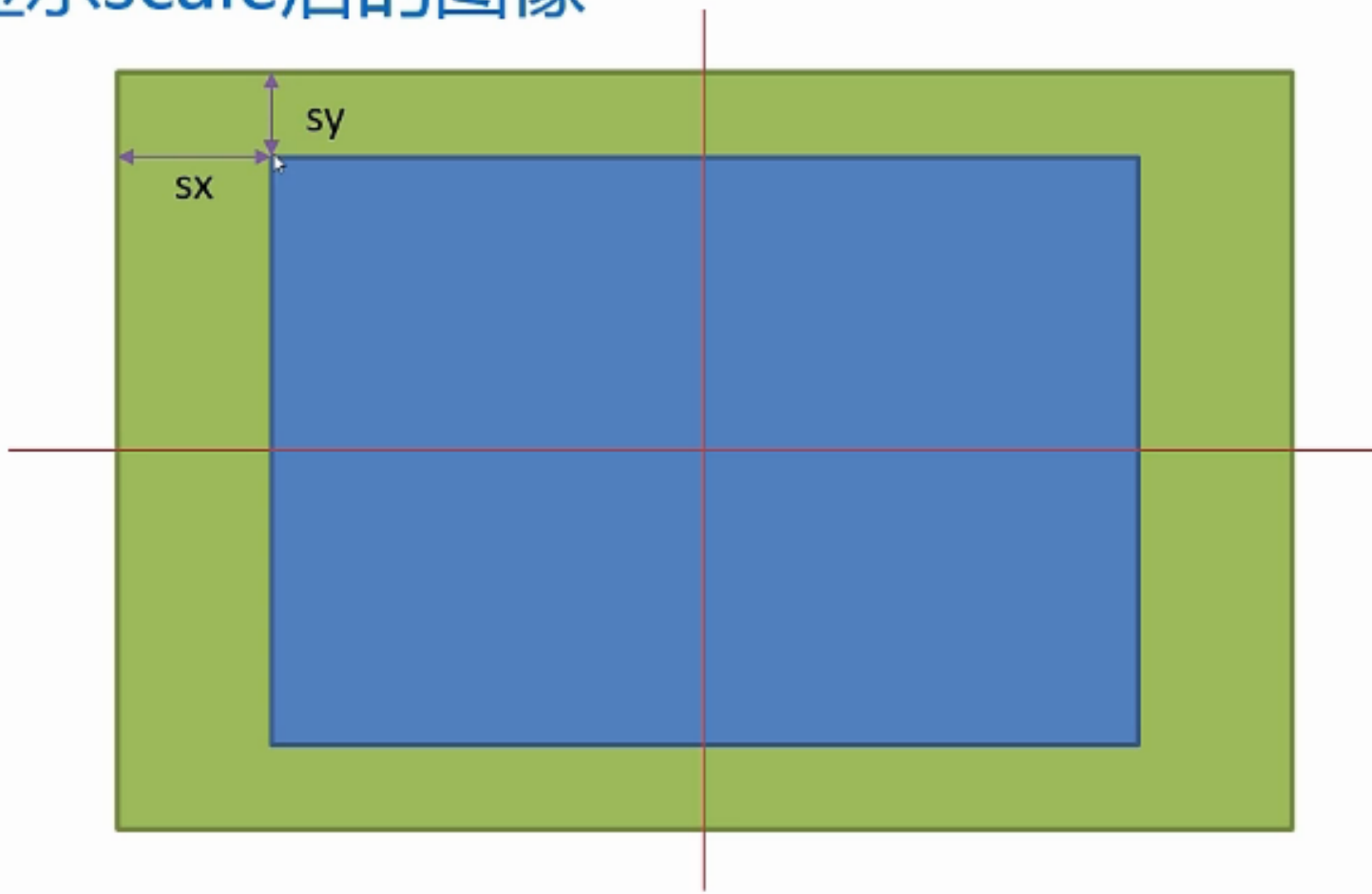
练习

使用滑动杆进行交互，在画布中心缩放图像



绘图实例分析

显示scale后的图像



绘制视频

- 🎨 drawImage() 方法在画布上绘制视频。
 - drawImage(video,x,y) 在画布上定位视频
 - drawImage(video,x,y,width,height) 规定视频的宽度和高度。
 - 仅对当前正在播放视频的帧进行描绘。

demo12-11.html

04

像素操作



像素

- 🎨 canvas 是逐像素进行渲染的。
- 🎨 canvas 中提供了强大的像素处理方法，据此可以对图片进行复杂的处理，包括改变图片透明度、图片反色、图片高亮、剪切、复制等操作。

ImageData

🎨 **ImageData** 对象，该对象为画布上指定的矩形像素数据。

- width — ImageData 对象的宽度，以像素计
- height — ImageData 对象的高度，以像素计
- data — ImageData 对象的**图像数据，数组类型**

🎨 ImageData 对象中的每个像素，都存在四方面的信息，即RGBA值。
数组中每四个数据代表一个像素的信息。

- R - 红色 (0-255) G - 绿色 (0-255) B - 蓝色 (0-255)
- A - alpha 通道 (0-255; 0 是透明的，255 是完全可见的)

创建ImageData对象


createImageData(width, height)

- 创建新的空白 ImageData 对象。新对象的默认像素值为 rgba(0,0,0,0)。

createImageData(imageData)

- 创建与指定的另一个 ImageData 对象尺寸相同的新 ImageData 对象，
但不会复制图像数据。

返回ImageData对象

 **getImageData**(x, y, width, height)

- 返回 ImageData 对象，该对象拷贝了画布指定矩形的像素数据。

 **putImageData**(imgData,x,y)

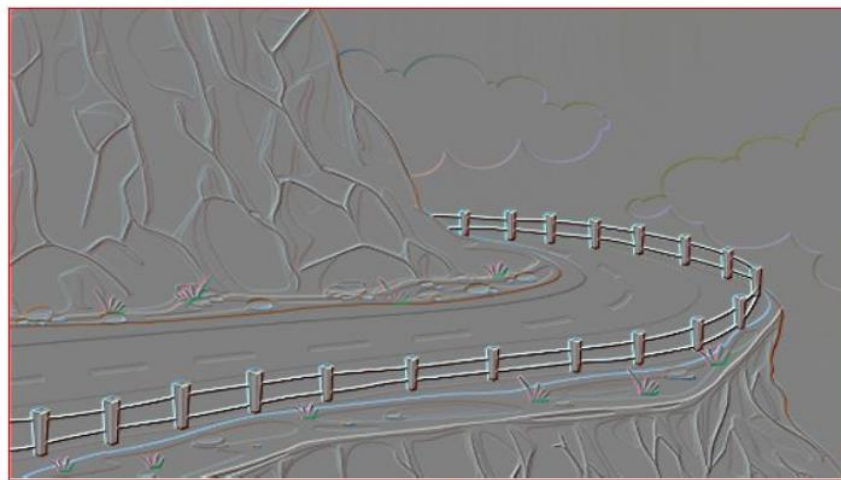
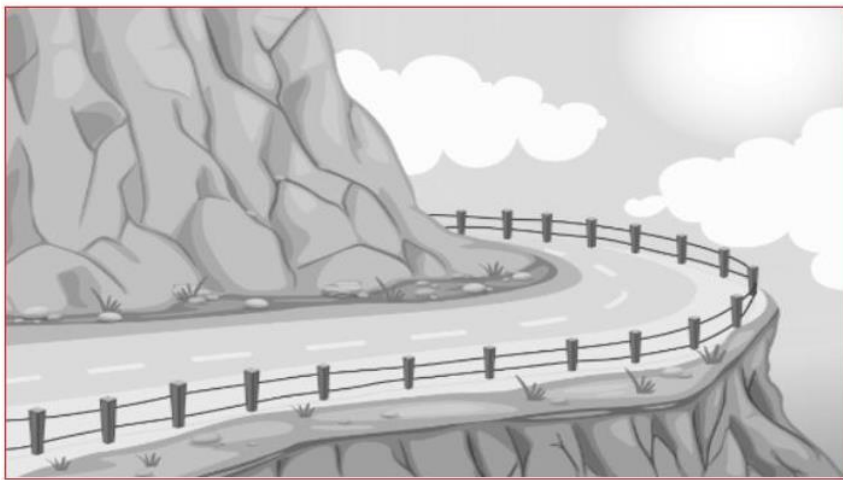
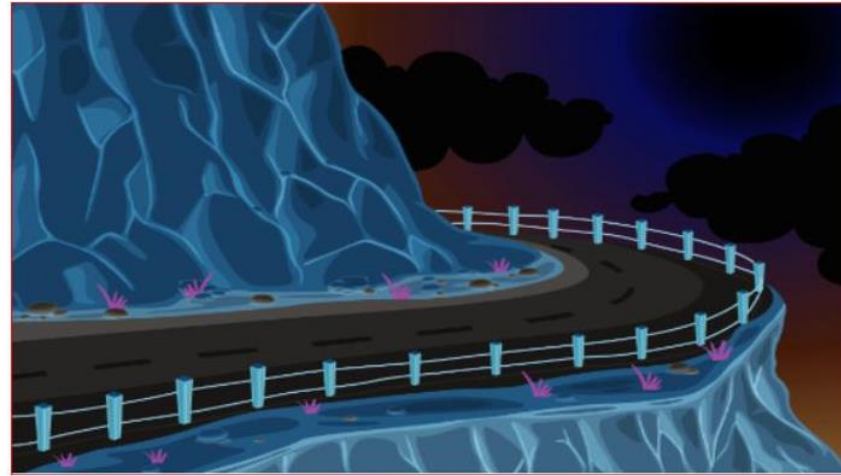
- 把指定的 ImageData 对象中的图像数据放到画布的指定位置上。

返回ImageData对象

putImageData(imgData,x,y)

参数	描述
imgData	规定要放回画布的 ImageData 对象。
x	ImageData 对象左上角的 x 坐标，以像素计。
y	ImageData 对象左上角的 y 坐标，以像素计。
dirtyX	可选。水平值（x），以像素计，在画布上放置图像的位置。
dirtyY	可选。水平值（y），以像素计，在画布上放置图像的位置。
dirtyWidth	可选。在画布上绘制图像所使用的宽度。
dirtyHeight	可选。在画布上绘制图像使用的高度。

实例



滤镜原理解释

- 1. 反色：获取一个像素点RGB值 r, g, b 。则新的RGB值为 $(255-r, 255-g, 255-b)$
- 2. 灰色调：获取一个像素点RGB值 r, g, b 。则新的RGB值为
$$\begin{aligned} \text{newr} &= (r * 0.272) + (g * 0.534) + (b * 0.131); \\ \text{newg} &= (r * 0.349) + (g * 0.686) + (b * 0.168); \\ \text{newb} &= (r * 0.393) + (g * 0.769) + (b * 0.189); \end{aligned}$$
- 3. 浮雕与雕刻：基于当前像素的前一个像素RGB值与它的后一个像素的RGB值之差再加上128

05

动画循环



动画循环

- ❖ 在 canvas 中实现动画的原理：在播放动画时持续更新并绘制动画，即动画循环。
- ❖ setInterval() 和 clearInterval()
 - 时间不精确，时间间隔不好把控
 - 动画不流畅
- ❖ requestAnimationFrame() 和 cancelAnimationFrame()
 - 浏览器自行决定播放最佳速度

动画循环

🎨 `window.requestAnimationFrame(callback)`

- 执行动画并请求浏览器在下一次重绘之前调用指定的函数来更新动画
- 需要下次重绘时，在回调函数中必须调用 `requestAnimationFrame ()`。

```
function animate() { //回调函数
    //动画内容
    //....

    requestAnimationFrame(animate);
}
```

demo12-14.html

THANKYOU

