5.5　利用双缓冲技术绘制动画

到目前为止，本章所用的动画逻辑都是下面这个样子的：

```
var canvas = document.getElementById('canvas'),
   context = canvas.getContext('2d'),
    ...
function animate(time) {
    context.clearRect(0, 0, canvas.width, canvas.height);
   // Update and draw animation objects...
   requestNextAnimationFrame(time); // Keep the animation going
}
   requestNextAnimationFrame(time); // Start the animation
```

上述代码首先清除 canvas，然后绘制下一帧动画。假如动画是单缓冲的（single buffered），那么就意味着其内容会被立刻绘制到屏幕 canvas 中。这样的话，擦除背景的那一瞬间所造成的空白可能会使动画看起来有些闪烁。

防止闪烁的一种办法就是使用双缓冲（double buffering）。如果用双缓冲，那么就不是将动画内容直接绘制到屏幕 canvas 中了，而是先将所有东西都绘制到离屏 canvas 里面，然后把该 canvas 的全部内容一次性地复制到屏幕 canvas 中。程序清单 5-13 演示了这种做法。

程序清单 5-13　利用双缓冲技术绘制动画

```
// For illustration only. Do not do this.

var canvas = document.getElementById('canvas'),
    context = canvas.getContext('2d'),

    // Create an offscreen canvas

    offscreenCanvas = document.createElement('canvas'),
    offscreenContext = offscreenCanvas.getContext('2d'),
    ...
offscreenCanvas.width = canvas.width;
offscreenCanvas.height = canvas.height;

function animate(now)  {
   offscreenContext.clearRect(
      0, 0, offscreenCanvas.width, offscreenCanvas.height);

      // Update and draw animation objects into the offscreen canvas...

      // Clear the onscreen canvas and draw the offscreen
      // into the onscreen canvas

      context.clearRect(0, 0, canvas.width, canvas.height);
      context.drawImage(offscreenCanvas, 0, 0);
}
```

双缓冲技术可以有效地消除动画绘制时的闪烁，所以浏览器会自动采用双缓冲来实现 canvas 元素。开发者并不需要自己来实现它。而且如果按照程序清单 5-13 所示方法来手工实现双缓冲的话，反倒会降低动画的绘制效率。在绘制每帧动画时，都花时间把离屏缓冲 canvas 的内容复制到屏幕上，这样做是毫无益处的，因为浏览器已经内建了对双缓冲技术的支持。

如果你曾经在调试器中单步执行过与 Canvas 有关的代码，那么可能会怀疑浏览器到底有没