JS动画

陈一帅

实务学堂

介绍

- Web动画API
- 用JavaScript写并控制动画

关键帧对象

- 首先创建一个对应于CSS @keyframes 的关键帧对象
  - 对象数组。每个对象代表原始CSS中的一个键
  - 默认等间隔，也可以指定偏移量

```
let movement = [
    { top: '-200px', left: '-200px' },
    { top: '480px', left: '480px' }
];
```

时间属性

- 持续时间是毫秒，不是秒：3000是3秒
- 重复次数是iterations, 不是iteration-count
- 无穷多次是Infinity，不是 “infinity”

```
// timing object
let skateTiming = {
    duration: 3000,
    iterations: Infinity
    };
```

整合这些特性

```
// put together with the animate method
let skate = skater.animate(
    movement,
    skateTiming
)
```

Example

MDN

操纵动画的播放

- skate.play() 开始
- skate.pause() 暂停
- skate.finish() 结束
- skate.cancel() 取消
- 倒退
  - 设置动画播放速度到负值，所以它向后运行
  - skate.playbackRate = -1
  - skate.reverse()

Example

仅指定持续时间

- 指定动画持续时间，而不是其重复次数（默认动画迭代一次）
- 毫秒

```
let skate = skater.animate(
    movement,
    3000
)
```

Example

更多动画属性

```
{
  fill: 'forwards',
  easing: 'steps(4, end)',
  duration:
    aliceChange.effect.timing.duration / 2
}
```

指定偏移量

- 用offset明确设置一个键与其他键的偏移

```
var aliceTumbling = [
  { transform: 'rotate(0)
        translate3D(-50%, -50%, 0)',
    color: '#000' },
  { color: '#431236', offset: 0.3},
  { transform: 'rotate(360deg)
        translate3D(-50%, -50%, 0)',
    color: '#000' }
];
```

设置CSS Style，进行动画控制

```
<button type="button" id="play">开始</button>
```

```
let playBtn = document.getElementById('play');

playBtn.addEventListener('click', play);

let cube = document.querySelector('.cube');

cube.style.animationPlayState = 'running';

cube.style.animationPlayState = 'paused';
```

Cubic控制

例：动画事件

```
function animationData(event) {
  if (event.type == "animationstart") {
    animationStatus.textContent = '动画开始啦';
  } else if (event.type == 'animationiteration') {
    iteration += 1;
    animationStatus.textContent = '动画重复次数: '
      + iteration;
  }
}
cube.addEventListener('animationstart',
  animationData);
cube.addEventListener('animationiteration',
  animationData);
```

Cubic控制

设置CSS Style 动画，进行动画控制

```
object.style.animation = "
  name 名称
  duration 时长
  timingFunction  定时
  delay 延时
  iterationCount  循环次数
  direction 方向
  fillMode 填充模式
  playState"  播放状态
```

W3School

设置CSS Style 过渡，进行动画控制

```
object.style.transition = "
  property 属性
  duration 时长
  timing-function delay|initial|inherit"
```

W3School

设置CSS Style 变换，进行动画控制

```
let xRotation = 60 - Math.ceil(yPos / yBrowserRatio);

squares[i].style.transform = 'rotateX(' + xRotation + 'de
```

- 网格形状的动态变换
  - 映射到光标移动
  - 缩放到浏览器窗口

例：鼠标位置控制旋转变换

请求动画帧，实现动画

- window.requestAnimationFrame()
  - 要求浏览器在下次重绘之前调用指定的回调函数更新动画
  - 传入回调函数，该函数会在浏览器下一次重绘之前执行
  - 动画 skate 函数会在浏览器准备重绘屏幕时运行

```
let animation = requestAnimationFrame(skate);
function skate() {
    animation = requestAnimationFrame(skate);
    skater.style.top = position + 'px';
```

Example

MDN

请求动画帧，实现动画

- skate 动画函数本身再次调用 requestAnimationFrame 安排下一次更新
  - 当浏览器窗口（或选项卡）处于活动状态时，这将导致更新以每秒约60的速度发生，这往往会产生很漂亮的动画
  - 通常是每秒执行60次（与浏览器屏幕刷新次数匹配）

```
let animation = requestAnimationFrame(skate);
function skate() {
    animation = requestAnimationFrame(skate);
    skater.style.top = position + 'px';
```

Example

MDN

为什么

- 为什么要用 requestAnimationFrame
  - 使浏览器知道我们现在已经完成，并且可以继续执行浏览器所要做的事情，例如更新屏幕和响应用户操作
  - 如果我们只是循环更新DOM，则页面将冻结，并且屏幕上不会显示任何内容。
  - 浏览器不会在JavaScript程序运行时更新其显示，也不允许与页面进行任何交互

好处

- 提升性能和电池寿命
  - 大多数浏览器里，当requestAnimationFrame() 运行在后台标签页或者隐藏的 iframe 里时，requestAnimationFrame() 会被暂停调用

取消

```
cancelAnimationFrame(animation);
```

- 取消一个先前通过调用window.requestAnimationFrame()方法添加到计划中的动画帧请求

MDN

例：以椭圆曲线移动图片

- 图片以页面为中心，相对位置
- 反复更新该图片的顶部和左侧样式以将其移动

```
let angle = Math.PI / 2;
function animate(time, lastTime) {
  if (lastTime != null) {
    angle += (time - lastTime) * 0.001; }
  cat.style.top = (Math.sin(angle) * 200) + "px";
  cat.style.left = (Math.cos(angle) * 200) + "px";
  requestAnimationFrame(
      newTime => animate(newTime, time));
}
```

Example

例：以椭圆曲线移动图片

- 回调函数的输入参数
  - 动画函数会被传入 DOMHighResTimeStamp 参数，该参数与 performance.now() 的返回值相同，它表示 requestAnimationFrame 执行回调函数的时刻
- 角度通过经过的时间计算，确保运动速度稳定

```
function animate(time, lastTime) {
  if (lastTime != null) {
    angle += (time - lastTime) * 0.001; }
  cat.style.top = (Math.sin(angle) * 200) + "px";
  requestAnimationFrame(
      newTime => animate(newTime, time));
```

Example

练习1

- 扩展先前定义的椭圆滑板动画
  - 加一个帽子图片，使运动员和帽子在椭圆上以相反方向旋转
  - 使帽子绕运动员转一圈
  - 以其他有趣的方式更改动画
- 提示
  - 为使定位多个对象更容易，最好切换到绝对定位
  - 这意味着顶部和左侧是相对于文档的左上角计算的
  - 为避免使用负坐标，这将导致图像移至可见页面之外。为此，可以向位置值添加固定数量的像素

提示

- Math.cos和Math.sin以弧度测量角度，其中整个圆为2π
- 对于给定的一个角度，您可以加π（即Math.PI），得到相反的角度
- 这对于将帽子放在轨道的另一侧可能很有用

练习2

- 访问 WDD 网站
- 选择你喜欢的作品
- 研究它的代码

练习3

- 从一个包含CSS过渡或动画的网页开始
  - 可以修改以前的项目，也可以开发新的项目
  - 添加Javascript动画，无需用户交互，就能够看到动画（类似滑板动画）
- 代码应至少包括两个DOM查询，一个定义的函数以及一个响应用户操作的事件侦听器
- 将所有文件放在新目录中并作为外部文档链接到CSS和JavaScript

练习4

- 使用JavaScript的requestAnimationFrame（）方法或Web动画API创建新的动画
- 提示
  - 如果使用requestAnimationFrame（）方法，则递归调用绘图函数，并随着时间的推移逐步更改CSS的某些方面
  - 如果使用Web动画API，则用animate（）方法将关键帧对象与计时对象组合在一起
  - 尽管此动画不需要响应用户的操作，但是您也可以自由地向此页面添加交互性