# CSS3

第3天课堂笔记（本课程共3天）

目录

[CSS3 1](#_Toc18419)

[目录 2](#_Toc27623)

[一、复习 3](#_Toc1372)

[二、圆周运动 4](#_Toc32251)

[三、作业中的小知识点 5](#_Toc1619)

[四、animation属性 6](#_Toc4895)

[4.1 整体感知 6](#_Toc17906)

[4.2 动画的定义 7](#_Toc451)

[4.3 动画的调用 7](#_Toc29089)

[五、velocity.js 8](#_Toc6020)

# 一、复习

CSS3中最最重要的属性：

● transition过渡，能够把元素的基本所有的属性变化，用动画的形式过渡出来；

|  |
| --- |
| 1. transition:all 1s ease 0s; |

css3中，所有的动画都是用transition实现（今天学习的aniamtion属性也能实现动画）。transition效率非常高，是底层C++在渲染动画，帧率非常高，非常平滑、细腻。比setInterval好一万倍。

● transform变形，有两大类2D变形，3D变形

旋转

|  |
| --- |
| 1. transform:rotate(50deg); |

斜切

|  |
| --- |
| 1. transform:skew(30deg,30deg); 2. 等价于： 3. transform:skewX(30deg); 4. transform:skewY(30deg); |

缩放

|  |
| --- |
| 1. transform:scale(2,1); 2. 等价于 3. transform:scaleX(2); 4. transform:scaleY(1); |

绕X、Y轴旋转

|  |
| --- |
| 1. transform:rotateX(30deg); 2. transform:rotateY(30deg); |

如果同时：

|  |
| --- |
| 1. transform:rotateX(30deg) rotateY(30deg); |

空间移动：

|  |
| --- |
| 1. transform:translateX(100px); 2. transform:translateY(200px); 3. transform:translateZ(300px); 4. 等价于 5. transform:translate3D(100px,200px,300px); |

还有一些杂碎属性：

|  |
| --- |
| 1. backface-visibility:hidden; 2. transform-origin:0% 0%; 3. transform-style:preserve-3d; |

# 二、圆周运动

先跑个题，讲讲圆周运动，工作中还比较有用。让一个元素，绕圆周进行运动。

当一个角度确定的时候，此时的x、y的坐标是什么：

|  |
| --- |
| 1. o.style.left = a + r \* Math.sin(degreeToRad(degree)) + "px"; 2. o.style.top = b - r \* Math.cos(degreeToRad(degree)) + "px"; |

一定要记住，Math.sin()、Math.cos()里面的值都是弧度制，而不是角度制。所以要进行转换：

|  |
| --- |
| 1. function degreeToRad(degree){ 2. return degree \* Math.PI / 180; 3. } |

看看案例。

# 三、作业中的小知识点

小知识点：

|  |
| --- |
| 1. transform:translateY(-50%); |

向上移动自己身高的一半。可以轻松实现绝对定位的盒子垂直居中。

就和我们之前说的：

|  |
| --- |
| 1. margin-top:-20px; |

差不多。

小知识点：

当一个元素是动态添加到DOM树的时候，transition不生效，需要时间准备。所以一定要记住：在创建元素之后，要把css的变化语句，放到1毫秒之后执行。

|  |
| --- |
| 1. //创建了一个div标签，追加到了body元素身上 2. $("<div></div>").appendTo("body"); 3. //改变css，我们预想的是，用过渡实现这个东西，但是是干蹦。 4. //因为浏览器没有反应过来，你可以认为transition需要准备时间。 5. setTimeout(function(){ 6. $("div").css("width",1000); 7. },**1**); |

# 四、animation属性

之前学习的transition能够实现动画，不过这个动画是需要触发条件的，比如css改变、类名的改变、hover了。

animation属性，可以让动画更加自由。

## 4.1 整体感知

必须先使用

|  |
| --- |
| 1. @-webkit-keyframes |

定义动画，然后使用animation属性调用动画

定义动画：

|  |
| --- |
| 1. **@-webkit-keyframes** yundong{ 2. from{ 3. left: 100px; 4. } 5. to{ 6. left:700px; 7. } 8. } |

@符号表示定义，之前我们学习过iconfont字体图片，我们也用过@符号：@font-family。你就当做死规定记住就行了，没什么大不了。keyframes是关键帧的意思，里面定义的不是每帧的变化情况，而是起点、终点。一会儿你将知道，还能定义一些动画过程中的点。

yundong就是我们任意取的动画的名字。

from、to从什么变为什么。

动画的调用，写在div的属性里面：

|  |
| --- |
| 1. animation: yundong 10s linear 0s; |

几个参数的意义我们一会儿说。

## 4.2 动画的定义

做简单的动画， 用from{} to{}。还能用百分比，就表示动画时间流逝了这么多的时候，应该所处的状态：

|  |
| --- |
| 1. @-webkit-keyframes yundong{ 2. 0%{ 3. left:100px; 4. } 5. 90%{ 6. left: 150px; 7. } 8. 100%{ 9. left: 700px; 10. } 11. } |

任何属性都能参与动画，transition能的，animation也能。

## 4.3 动画的调用

animation属性完整的调用一共有7个参数：

|  |
| --- |
| 1. animation: yundong 1s linear 0s 5 alternate forwards; |

第1个参数： 动画的名字

第2个参数： 动画的总时长，别忘了写s

第3个参数： 缓冲

第4个参数： 延迟时间，注意如果动画进行很多次，只有第一次开始前有延迟

第5个参数： 运动次数，要无限次写infinite

第6个参数： 是否来回运动，如果写alternate那么第偶数次将是时间轴的逆向

第7个参数： forwards表示当运动结束之后让这个元素保留在原地

# 五、velocity.js

库有自己的哲学，就是解决了什么样子的用户痛点。凭什么要用这个，用这个有什么好处？



velocity.js应运而生，是一个动画引擎，它能够独立于jQuery运行，但是自己不具备$()选择元素功能；它又能够给所有的jQuery对象加上一个方法叫做velocity方法。也就是说，velocity.js可以认为是jQuery插件。

官网： <http://www.julian.com/research/velocity/>

velocity给jQuery对象增加了velocity方法，API和jQuery中的animate函数一样一样的：

|  |
| --- |
| 1. $("img").velocity({"width":400,"height":400},1000,function(){ 2. $(this).hide(); 3. }); |

通过观察，知道了velocity的动画机理还是定时器，底层还是setInterval那个运动框架。也就是说在动画机理层面上，velocity并没有做出质的改变。

那velocity有什么意义呢？它拓展了jQuery中的animate函数。

velocity能够实现rotateZ的动画：

|  |
| --- |
| 1. $("img").velocity({"rotateZ":"40deg"},1000); |

机理是定时器！！

velocity能够实现背景颜色的动画：

|  |
| --- |
| 1. $("body").velocity({"backgroundColor":"#123"},10000); |

机理是定时器！！

velocity能够实现3D的动画：

|  |
| --- |
| 1. $("img").velocity({"rotateX":"40deg"},1000); |

机理是定时器！！

总结一下：

velocity放弃了所有能够用过渡实现的动画效果。连rotateX这种3D的变化，都用定时器实现了。所以感觉是一种“退化”，明明用transition能实现，为什么还要用定时器呢？毕竟我们之前说过，transition的效率比定时器高过了。这就叫做“舍得”，有舍才有得，velocity放弃了效率，但是赢回了“正向编程”编程的快感。你会发现，动画时间不用写在css里面了，也有了回调函数。

一定要记住velocity就是jQuery的插件，丰富了jQuery中animate函数的功能。

|  |
| --- |
| 1. $("img").velocity({ 2. "rotateX":"40deg", 3. "rotateY":"40deg", 4. "translateZ":"400px" 5. },1000,function(){ 6. //回调函数 7. $(this).fadeOut(); 8. }); |