【CSS3】transition过渡和animation动画

写在前面的话：

之前实习的时候，刚开始的第一个月就是在研究CSS3的动画，因为要做转盘抽奖活动，预研的时候，我是用Canvas来画的，当时为了一个问题“如何使用canvas让图片围绕中心点旋转”折腾了半天啊，最后好在是解决了,我可是google了很多，看了很多stackoverflow上的英文才弄明白，==我是不是有点傻……（**如果有感兴趣的同学想要尝试用Canvas让图片围绕中心点旋转，可以私信我，我给你发代码呀，哈哈哈**）但是带我的师父给我说，你不觉着这个实现起来有些繁琐吗，你再想想，让转盘旋转还能用什么方法？CSS3动画可以不可以？所以，我就开始研究CSS3动画，用CSS3来让图片旋转，真的好简单啊，简单到要哭了。可是，郁闷的是，虽然那段时间经常在用CSS3动画，可是又几个月过去了，竟然变得生疏了。所以，内推之前又开始复习了一遍，多总结。

正文开始

本来只想总结animation，但是转念一想，该先看看transition的，毕竟都跟动画有关系吼。那么先来说一下transition。

一、transition

CSS3的过渡功能就像是一种黄油，可以让CSS的一些变化变得平滑。因为原生的CSS过渡在客户端需要处理的资源要比用JavaScript和Flash少的多，所以才会更平滑。

transition的属性

属性	描述
transition - property	指定要过渡的 CSS 属性
transition - duration	指定完成过渡要花费的时间
transition - timing-function	指定过渡函数
transition - delay	指定过渡开始出现的延迟时间

属性可以分开写，也可以放在一起写,比如下面的代码，图片的宽高本来都是15px，想要让它1秒的时间内过渡到宽高为450px，通过：hover来触发，那么代码就可以如下：

```
img{
    height:15px;
    width:15px;
    transition: 1s 1s height ease;/*合在一起*/
}
或者：
img{
    height：15px;
    width: 15px;
    transition-property: height;
    transition-duration: 1s;
    transition-delay: 1s;
    transition-timing-function: ease;/*属性分开写*/
}
img:hover{
    height: 450px;
    width: 450px;
}
```

因为过渡所需要时间与过渡延迟时间的单位都是秒，所以在合起来写transition的属性的时候，第一个time会解析为transiton-duration，第二个解析为transition-delay。**所以，可以给transition一个速记法**

transiton: 过渡属性 过渡所需要时间 过渡动画函数 过渡延迟时间；

属性详解

transition-property

不是所有属性都能过渡，只有属性具有一个中间点值才具备过渡效果。完整列表，见这个列表，具体效果，见 这个页面。

transition-duration

指定从一个属性到另一个属性过渡所要花费的时间。默认值为0，为0时，表示变化是瞬时的，看不到过渡效果。

transiton-timing-function

过渡函数，有如下几种：

liner：匀速

ease-in：减速

ease-out：加速

ease-in-out：先加速再减速

cubic-bezier：三次贝塞尔曲线，可以定制 点击这里

触发过渡

单纯的代码不会触发任何过渡操作，需要通过用户的行为（如点击，悬浮等）触发，可触发的方式有：

:hoever :focus :checked 媒体查询触发 JavaScript触发

局限性

transition的优点在于简单易用，但是它有几个很大的局限。

（1）transition需要事件触发，所以没法在网页加载时自动发生。

（2）transition是一次性的，不能重复发生，除非一再触发。

（3）transition只能定义开始状态和结束状态，不能定义中间状态，也就是说只有两个状态。

（4）一条transition规则，只能定义一个属性的变化，不能涉及多个属性。

CSS Animation就是为了解决这些问题而提出的。

二、animation

CSS3的animation属性可以像Flash制作动画一样，通过控制关键帧来控制动画的每一步，实现更为复杂的动画效果。ainimation实现动画效果主要由两部分组成：

1）通过类似Flash动画中的帧来声明一个动画；

2）在animation属性中调用关键帧声明的动画。

Note： animation属性到目前位置得到了大多数浏览器的支持，但是，需要添加浏览器前缀哦！需要添加浏览器前缀哦！需要添加浏览器前缀哦！

animation动画属性

还是先列表格来说明属性，自己感觉会比较清晰：

属性	描述
animation-name	用来指定关键帧动画的名字
animation-duration	用来指定动画播放所需的时间，一般以秒为单位
animation-timing-function	设置动画的播放方式
animation-delay	指定动画的开始时间，以秒为单位
animation-iteration-count	指定动画播放的循环次数
animation-direction	控制动画的播放方向
animation-play-state	设置动画的播放状态，播放还是暂停
animation-fill-mode	设置动画的时间外属性

（1）animation-name：none为默认值，将没有任何动画效果，其可以用来覆盖任何动画

（2）animation-duration：默认值为0，意味着动画周期为0，也就是没有任何动画效果

（3）animation-timing-function：与transition-timing-function一样

（4）animation-delay：在开始执行动画时需要等待的时间

（5）animation-iteration-count：定义动画的播放次数，默认为1，如果为infinite，则无限次循环播放

（6）animation-direction：默认为nomal，每次循环都是向前播放，（0-100），另一个值为alternate，动画播放为偶数次则向前播放，如果为基数词就反方向播放

（7）animation-state：默认为running，播放，paused，暂停

（8）animation-fill-mode：定义动画开始之前和结束之后发生的操作，默认值为none，动画结束时回到动画没开始时的状态；forwards，动画结束后继续应用最后关键帧的位置，即保存在结束状态；backwards，让动画回到第一帧的状态；both：轮流应用forwards和backwards规则。

@keyframes

CSS3的animation制作动画效果主要包括两部分：1. 用关键帧声明一个动画，2.在animation调用关键帧声明的的动画。

@keyframes就是关键帧。这个帧与Flash里的帧类似，一个动画中可以有很多个帧。

一个@keyframes中的样式规则是由多个百分比构成的，可以在这个规则上创建多个百分比，从而达到一种不断变化的效果。另外，**@keyframes必须要加webkit前缀**。举个例子：

```
div:hover {
  -webkit-animation: 1s changeColor;
  animation: 1s changeColor;
}

@-webkit-keyframes changeColor {
  0% { background: #c00; }
  50% { background: orange; }
  100% { background: yellowgreen; }
}
```

```
@keyframes changeColor {
  0% { background: #c00; }
  50% { background: orange; }
  100% { background: yellowgreen; }
}
```

上面代码中的0% 100%的百分号都不能省略，0%可以由from代替，100%可以由to代替。要让changeColor动画有效果，就必须要通过CSS3animation属性来调用它。

区别

animation属性类似于transition，他们都是随着时间改变元素的属性值，其主要区别在于：transition需要触发一个事件才会随着时间改变其CSS属性；animation在不需要触发任何事件的情况下，也可以显式的随时间变化来改变元素CSS属性，达到一种动画的效果
我做了一个小例子，使用animation来模拟PPT播放，实现了从左边进，从上面进，放大，缩小和旋转。地址：这里
过程中，又学习和巩固了一些CSS3的新特性，一并总结在这里：

伪类选择器

目标伪类选择器

：target，目标伪类选择器，用来匹配文档的URI中某个标识符的目标集。具体来说，URI中的标识符通常会包含一个#，后面带有一个标识符名称。例如

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>:target</title>
    <style type="text/css">
        section{
            width: 500px;
            height: 140px;
            margin: 30px auto;
        }
        a{
            text-decoration: none;
            color: #828282;
            display: block;
            text-shadow: -1px -1px 1px rgba(0,0,0,0.8),
                         -2px -2px 1px rgba(0,0,0,0.3),
                         -3px -3px 1px rgba(0,0,0,0.3);
            text-align: center;
        }
        div {
            border: 1px solid #000;
            height: 100px;
            width: 500px;
            margin:30px auto 0 auto;
            padding: 10px;
        }
        :target{
            border: 2px solid #D4D4D4;
            background-color: #e5eecc;
        }
    </style>
</head>
<body>
    <section>
        <a href="#contact">联系我</a>
        <div id="contact">
            可通过私信联系我或者给我发邮件
            simplywenjing@163.com
        </div>
    </section>
</body>
</html>
```

：target前面如果什么都不加，就是匹配页面上所有的URL里的# 。
“：target”伪类选择器，选取链接的目标元素，然后定义样式。目标伪类选择器是动态选择器，只有存在URL指向该匹配元素时，样式效果才会生效。如上面代码中，div：target 才会生效，如果写的是p:target则不会产生任何效果，因为页面中没有p元素。 这里通过target伪类实

现了手风琴效果，详情请见github

否定伪类选择器

：not（）类似jQuery中的：not（）选择器，主要用来定位不匹配该选择器的元素。其用途还是很强大的，举2个例子：

```
:not(footer){
    border:1px solid black;/*表示选择页面中所有元素，除了footer*/
}
input:not([type=submit]){
    ... /*给表单中所有input定义样式，除了submit按钮*/
}
```

:nth-of-type

:nth-of-type与:nth-child的区别，举个例子来解释：

```
<div class="post">
    <p>第一个段落</p>
    <p>第二个段落</p>
</div>
```

接下来，我们想给第二个段落改变字体颜色，改成红色

```
.post>p:nth-child(2){color:red}
.post>p:nth-of-type(2){color:red}
```

这两段代码都可以让第二段变成红色，但这并不是说这两个选择器就是一样的。接着看，如果页面结构变成这样

```
<div class="post">
    <h1>标题</h1>
    <h2>副标题</h2>
    <p>第一个段落</p>
    <p>第二个段落</p>
</div>
```

再用上面的样式，你会发现

```
.post>p:nth-child(2){color:red}/*没有元素变红色*/
.post>p:nth-of-type(2){color:red}/*还是第二个段落变红色*/
```

添加了h1后，:nth-child(2)还是在选择div的第二个子元素，而这时候，div的第二个子元素是h2，与p不匹配，所以p:nth-child(2)匹配不到任何元素。

:nth-child( )选择的是某父元素的子元素，这个子元素并没有确切的类型；:nth-of-type( )选择的某父元素的某子元素，这个子元素是指定的类型

文本阴影

还用到了text-shadow属性，前面的例子里也使用到了，

```
text-shadow: -1px -1px 1px rgba(0,0,0,0.9);/*X轴位移，Y轴位移，阴影模糊半径，颜色*/
```

我觉着，CSS还是很强大，学好JavaScript固然重要，但是也不能把CSS丢在一边，我个人还是很喜欢学习CSS的，很酷炫啊。