前端深入之css篇丨2020年，彻底掌握css动画【animation】

写在前面

已经2020年了，不知道小伙伴们今年学习了css3动画了吗?

说起来css动画是一个很尬的事，一方面因为公司用css动画比较少，另一方面大部分开发者习惯了用JavaScript来做动画，所以就导致了许多程序员比较排斥来学习css动画（至少我是），但是一个不懂css动画的前端工程师不能称之为掌握css3，其实当你真正学习css动画之后，你会被它的魅力所吸引的，它可以减少代码量、提高性能。

这是一个系列文章，共分为四篇。

前端深入之css篇丨2020年，彻底掌握css动画【transition】

前端深入之css篇丨2020年，彻底掌握css动画【animation】

前端深入之css篇丨2020年，彻底掌握css动画【transform】

前端深入之css篇丨初探【transform】，手把手带你实现1024程序员节动画

这一文章的前置知识为**transition**，如果有不了解的同学可以点击对应链接学习。

话不多说，马上和我一起去学习今天的主角**animation**吧!

animation 语法

值	描述
@keyframes	定义一个动画,@keyframes定义的动画名称用来被animation-name所使用
animation-name	检索或设置对象所应用的动画名称 ,必须与规则@keyframes配合使用，因为动画名称由@keyframes定义
animation-duration	检索或设置对象动画的持续时间
animation-timing-function	检索或设置对象动画的过渡类型
animation-delay	检索或设置对象动画的延迟时间
animation-iteration-count	检索或设置对象动画的循环次数
animation-direction	检索或设置对象动画在循环中是否反向运动
animation-play-state	检索或设置对象动画的状态

> animation翻译成中文是动画的意思，熟练运用之后你可以用它来做各种各样炫酷的动画。

@keyframes：定义一个动画，定义的动画名称用来被animation-name所使用。


```
div{
    width:50px;
    height:50px;
    background:#f40;
    border-radius:50%;
    animation:mymove 2s;
}

@keyframes mymove{
    0%   {width:50px;height:50px;}
    50%   {width:100px;height:100px;}
    100%   {width:50px;height:50px;}
}
复制代码
```

@keyframes主要是做关键帧动画的，每个@keyframes后面都要跟一个名字，事例中使用了`mymove`作为帧动画的名字，然后可以在样式内对关键帧添加样式，然后根据关键帧`@keyframes`就能自动形成流畅的动画了。

animation-name：检索或设置对象所应用的动画名称 ,必须与规则@keyframes配合使用，因为动画名称由@keyframes定义


```
div{
    width:50px;
```

```
    height:50px;
    background:#f40;
    border-radius:50%;
    animation-name:mymove;
    animation-duration:2s;
}

@keyframes mymove{
    0%   {width:50px;height:50px;}
    50%   {width:100px;height:100px;}
    100%   {width:50px;height:50px;}
}
复制代码
```

在`animation-name`使用之前，我们已经定义了一个名为`mymove`的帧动画，这里把帧动画的名字作为了animation-name的值，含义是当前元素将执行所设置的帧动画。

animation-duration：检索或设置对象动画的持续时间

继续看上一个案例，仅仅有帧动画和需要执行的动画名称是不足以形成动画的，我们还需要设置一个动画执行所需要的时间，这里就用到了`animation-duration`属性，所以上一案例所展示的时间为两秒钟执行一次。

animation-timing-function：检索或设置对象动画的过渡类型


```
div{
    width:100px;
    height:50px;
    background:#f40;
    position:relative;
    animation-name:mymove;
    animation-duration:3s;
    animation-timing-function:ease-in-out;
}
@keyframes mymove{
    0% {left:0px;}
    100% {left:300px;}
}
复制代码
```

`animation-timing-function`的作用就是改变动画在每一帧的快慢。这里`transition-timing-function`仅展示值为`ease-in-out`的动画效果，可以理解为`慢-快-慢`。其他的不做展示，可以参考下表进行理解。

值	描述
linear	动画从头到尾的速度是相同的。
ease	默认。动画以低速开始，然后加快，在结束前变慢。
ease-in	动画以低速开始。
ease-out	动画以低速结束。
ease-in-out	动画以低速开始和结束。
cubic-bezier(n,n,n,n)	在 cubic-bezier 函数中自己的值。可能的值是从 0 到 1 的数值。

animation-delay：检索或设置对象动画的延迟时间


```
div{
    width:50px;
    height:50px;
    background:#f40;
    border-radius:50%;
    animation-name:mymove;
    animation-duration:2s;
    animation-delay:2s;
}

@keyframes mymove{
    0%   {width:50px;height:50px;}
    50%   {width:100px;height:100px;}
    100%   {width:50px;height:50px;}
}
复制代码
```

这里`animation-delay`的值为`2s`，意思是动画将在延迟两秒秒后延迟执行。

animation-iteration-count：检索或设置对象动画的循环次数


```
div{
    width:50px;
    height:50px;
    background:#f40;
    border-radius:50%;
    animation-name:mymove;
    animation-duration:2s;
    animation-iteration-count:infinite;
}

@keyframes mymove{
    0%   {width:50px;height:50px;}
    50%   {width:100px;height:100px;}
    100%   {width:50px;height:50px;}
}
复制代码
```

这里`animation-iteration-count`的值为`infinite`，意思是动画将会无限次的执行，这也就达到了循环的效果，当然你还可以给它具体的数值，当执行你设置的次数后它会自动停止。

animation-direction：检索或设置对象动画在循环中是否反向运动


```
div{
    width:100px;
    height:50px;
    background:#f40;
    position:relative;
    animation-name:mymove;
    animation-duration:2s;
    animation-iteration-count:infinite;
```

```
    animation-direction:alternate;
}

@keyframes mymove{
    0% {left:0px;}
    100% {left:300px;}
}
复制代码
```

这里`animation-direction`的值为`alternate`，代表动画将会来回的反复执行，他还有其它属性，在下表给出供小伙伴们自己尝试。

值	描述
normal	默认值。动画按正常播放。
reverse	动画反向播放。
alternate	动画在奇数次（1、3、5...）正向播放，在偶数次（2、4、6...）反向播放。
alternate-reverse	动画在奇数次（1、3、5...）反向播放，在偶数次（2、4、6...）正向播放。
initial	设置该属性为它的默认值。
inherit	从父元素继承该属性。

animation-play-state：检索或设置对象动画的状态


```
<style>
div{
    width:50px;
    height:50px;
    background:#f40;
    border-radius:50%;
    animation-name:mymove;
    animation-duration:2s;
    animation-iteration-count:infinite;
```

```
}
@keyframes mymove{
    0%   {width:50px;height:50px;}
    50%   {width:100px;height:100px;}
    100%   {width:50px;height:50px;}
}
</style>
<body>
    <button onclick="pause()">暂停</button>
    <button onclick="run()">恢复</button>
    <div></div>
</body>
复制代码
function pause(){
    document.querySelector('div').style.animationPlayState="paused"
}
function run(){
    document.querySelector('div').style.animationPlayState="running"
}
复制代码
```

`animation-play-state` 的默认值为 `running`，就是动画执行的意思，在实际应用中我们经常使用js来操作这个属性，从而控制动画的播放和暂停。

今天我们一共学习了八个属性值，他们都是属于 `animation` 属性的，这里给出属性值在 `animation` 中的简写方式。

> animation: name duration timing-function delay iteration-count direction play-state;

```
div{
    animation:mymove 2s ease-in-out 3s infinite alternate running;
}
复制代码
```

那么这里的意思就是 `mymove` 动画将在三秒钟后开始，以两秒一个循环慢-快-慢方式，进行动画的展示，并且每次动画过后都会向相反方向执行动画。

结论

经过以上的学习，相信你已经初步了解了 `animation` 的用法，随着你对 `animation` 的深入理解，是可以做一些有创造性的动画的，你可以看看自己之前用js写的各种动画，尝试着用我们两篇文章所学的内容进行修改，相信你一定会对这两个属性有更深的理解。

但是现在我们只是学会了过渡和动画，我们现在还不能对图形进行一系列的不规则操作，而 `transform（变形）` 就是来操作改变成特殊图形的，我将在接下来的文章继续为你讲解 `translate（移动）` 以及 `transform（变形）`，跟进我的脚步吧，跟我一起在2020年掌握css动画！