专题六: 彻底弄懂前端动画世界

> 大纲目录:
> - 动画细节和原理深入解析
> - transform 全解
> - transition 过渡
> - animation 动画与交互
> - 关键帧动画steps功能符深入介绍

动画细节和原理深入解析

> 动画的原理:
> - 视觉暂留作用
> - 画面逐渐变化
>
> 动画的作用:
> - 愉悦感
> - 引起注意
> - 掩饰
>
> 动画的类型:
> - transition补间动画
>   - 位置 – 平移(left/right/margin/transform：translate)
>   - 方位 – 旋转(transform：rotate)
>   - 大小 – 缩放(transform：scale)
>   - 透明度(opacity)
>   - 其他 – 线性变换(transform)
> - keyframes关键帧动画
>   - 相当于多个补间动画
>   - 与元素状态的变化无关

    - 定义更加灵活
- 逐帧动画
    - 适用于无法补间计算的动画
    - 资源较大
    - 使用steps()

transform 全解

translate(*x,y*)	定义 2D 转换，沿着 X 和 Y 轴移动元素。
translateX(*n*)	定义 2D 转换，沿着 X 轴移动元素。
translateY(*n*)	定义 2D 转换，沿着 Y 轴移动元素。
scale(*x,y*)	定义 2D 缩放转换，改变元素的宽度和高度。
scaleX(*n*)	定义 2D 缩放转换，改变元素的宽度。
scaleY(*n*)	定义 2D 缩放转换，改变元素的高度。
rotate(*angle*)	定义 2D 旋转，在参数中规定角度。

translate位移

- translate()： 指定对象的2D translation（2D平移）。第一个参数对应X轴，第二个参数对应Y轴。如果第二个参数未提供，则默认值为0
- translateX()： 定对象X轴（水平方向）的平移
- translateY()：指定对象Y轴（垂直方向）的平移

rotate旋转

- rotate()： 指定对象的2D rotation（2D旋转），需先有 <’transform-origin‘> 属性的定义；表示旋转一定的角度；
- rotate() = rotate(angle)
- rotate3d() = rotate3d(number,number,number,angle)
- rotateX() = rotatex(angle)
- rotateY() = rotatey(angle)
- rotateZ() = rotatez(angle)

scale缩放

- scale()：指定对象的2D scale（2D缩放）。第一个参数对应X轴，第二个参数对应Y轴。如果第二个参数未提供，则默认取第一个参数的值；
- scaleX()：指定对象X轴的（水平方向）缩放；
- scaleY()：指定对象Y轴的（垂直方向）缩放；

transform-origin元素变换基点

- 设置或检索对象以某个原点进行转换。

- 该属性提供2个参数值。
  - 如果提供两个，第一个用于横坐标，第二个用于纵坐标。
  - 如果只提供一个，该值将用于横坐标；纵坐标将默认为50%;
  - 默认值：50% 50%，效果等同于center center
- percentage： 用百分比指定坐标值。可以为负值。
- length： 用长度值指定坐标值。可以为负值。
- left： 指定原点的横坐标为left
- center①： 指定原点的横坐标为center
- right： 指定原点的横坐标为right
- top： 指定原点的纵坐标为top
- center②： 指定原点的纵坐标为center
- bottom： 指定原点的纵坐标为bottom

transform-3D

- 3D变换：在3D空间中进行变换
- 3D卡片
- 3D相册
- 3D立方体(实战)

```
// CSS
.container{
    margin:50px;
    padding: 10px;
    border: 1px solid red;
    width: 200px;
    height: 200px;
    position: relative;
    perspective: 500px;
}
#cube{
    width:200px;
    height:200px;
    transform-style: preserve-3d;
    transform: translateZ(-100px);
    transition:transform .4s;
}
#cube div{
    width: 200px;
    height:200px;
```

```
    position: absolute;
    line-height: 200px;
    font-size:50px;
    text-align: center;
}
#cube:hover{
    transform: translateZ(-100px) rotateX(90deg) rotateY(90deg);
}
.front{
    transform: translateZ(100px);
    background:rgba(255,0,0,.3);
}
.back{
    transform: translateZ(-100px) rotateY(180deg);
    background:rgba(0,255,0,.3);
}
.left{
    transform: translateX(-100px) rotateY(-90deg);
    background:rgba(0,0,255,.3);
}
.right{
    transform: translateX(100px) rotateY(90deg);
    background:rgba(255,255,0,.3);
}
.top{
    transform: translateY(-100px) rotateX(-90deg);
    background:rgba(255,0,255,.3);
}
.bottom{
    transform: translateY(100px) rotateX(90deg);
    background:rgba(0,255,255,.3);
}
// HTML
<div class="container">
  <div id="cube">
      <div class="front">1</div>
      <div class="back">2</div>
      <div class="right">3</div>
      <div class="left">4</div>
      <div class="top">5</div>
```

```
61        <div class="bottom">6</div>
62    </div>
63 </div>
```

transition 过渡

属性	描述	CSS
transition	简写属性，用于在一个属性中设置四个过渡属性。	3
transition-property	规定应用过渡的 CSS 属性的名称。	3
transition-duration	定义过渡效果花费的时间。默认是 0。	3
transition-timing-function	规定过渡效果的时间曲线。默认是 "ease"。	3
transition-delay	规定过渡效果何时开始。默认是 0。	3

复合属性。检索或设置对象变换时的过渡。

注意：

- 如果只提供一个参数，则为 <'transition-duration'> 的值定义；
- 如果提供二个参数，则第一个为 <' transition-duration '> 的值定义，第二个为 <' transition-delay'> 的值定义
- 可以为同一元素的多个属性定义过渡效果,如果需要定义多个过渡属性且不想指定具体是哪些属性过渡,可以用all代替;

transition-property:

- 检索或设置对象中的参与过渡的属性。
- 默认值为：all。默认为所有可以进行过渡的css属性。
- 如果提供多个属性值，以逗号进行分隔。
  - none： 不指定过渡的css属性
  - all： 所有可以进行过渡的css属性
  - <IDENT>: 指定要进行过渡的css属性

transition-duration:

- 检索或设置对象过渡的持续时间,默认值：0；
- 如果提供多个属性值，以逗号进行分隔；
- 取值: time 指定对象过渡的持续时间

transition-timing-function :

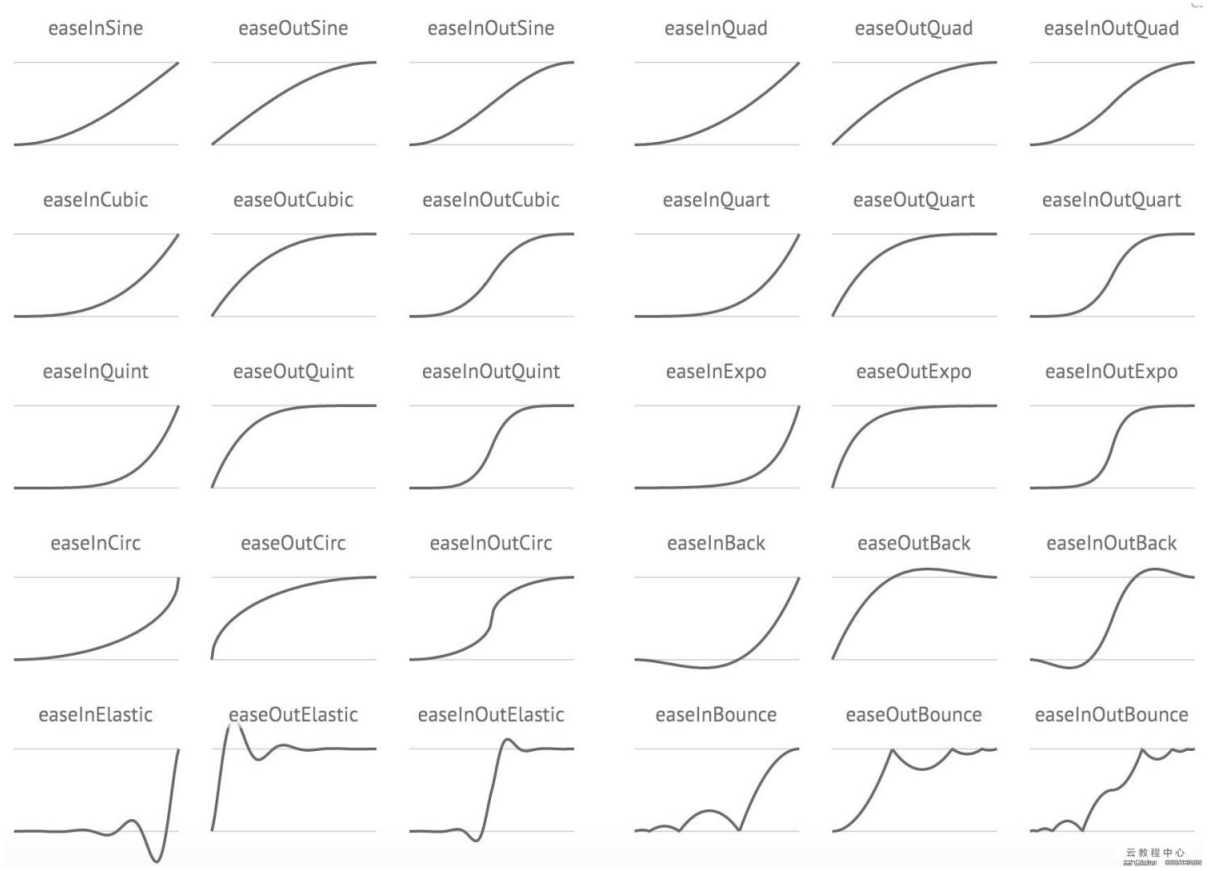
检索或设置对象中过渡的动画类型，默认值：ease；

如果提供多个属性值，以逗号进行分隔。

- linear： 线性过渡。等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0)

- ease： 平滑过渡。等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0)
- ease-in： 由慢到快。等同于贝塞尔曲线(0.42, 0, 1.0, 1.0)
- ease-out： 由快到慢。等同于贝塞尔曲线(0, 0, 0.58, 1.0)
- ease-in-out： 由慢到快再到慢。等同于贝塞尔曲线(0.42, 0, 0.58, 1.0)
- step-start： 等同于 steps(1, start)
- step-end： 等同于 steps(1, end)
- steps(<integer>[, [start | end]]?)： 接受两个参数的步进函数。第一个参数必须为正整数，指定函数的步数。第二个参数取值可以是start或end，指定每一步的值发生变化的时间点。第二个参数是可选的，默认值为end。
- cubic-bezier(<number>, <number>, <number>, <number>)： 特定的贝塞尔曲线类型，4个数值需在[0, 1]区间内

定义动画进度和时间的关系:
https://matthewlein.com/tools/ceaser
https://cubic-bezier.com/#.17,.67,.83,.67 贝塞尔曲线



transition-delay:

- 检索或设置对象延迟过渡的时间,，默认值：0；
- 如果提供多个属性值，以逗号进行分隔；
- time：指定对象过渡的延迟时间

animation 动画与交互

属性	描述	CSS
@keyframes	规定动画。	3
animation	所有动画属性的简写属性，除了 animation-play-state 属性。	3
animation-name	规定 @keyframes 动画的名称。	3
animation-duration	规定动画完成一个周期所花费的秒或毫秒。默认是 0。	3
animation-timing-function	规定动画的速度曲线。默认是 "ease"。	3
animation-delay	规定动画何时开始。默认是 0。	3
animation-iteration-count	规定动画被播放的次数。默认是 1。	3
animation-direction	规定动画是否在下一周期逆向地播放。默认是 "normal"。	3
animation-play-state	规定动画是否正在运行或暂停。默认是 "running"。	3
animation-fill-mode	规定对象动画时间之外的状态。	3

CSS动画库: https://daneden.github.io/animate.css/

@keyframes 动画帧

- 指定动画名称和动画效果。
- @keyframes定义的动画名称用来被animation-name所使用。
- 定义动画时，简单的动画可以直接使用关键字from和to，即从一种状态过渡到另一种状态：

语法: @keyframes<identifier> { <keyframes-blocks> }

<keyframes-blocks>: [[from | to | <percentage>]{ sRules }] [[, from | to | <percentage>] { sRules }]*

取值: < identifier>: identifier定义一个动画名称

<keyframes-blocks> : 定义动画在每个阶段的样式，即帧动画

```
@keyframes test {
  0% { opacity: 1; }
  50% { opacity: .5; }
  100% { opacity: 0; }
}
```

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

- none: 不引用任何动画名称
- identifier: 定义一个或多个动画名称(identifier标识)

animation-duration:

检索或设置对象动画的持续时间,默认值：0s

- time 指定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，默认值：ease；

- linear： 线性过渡。等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0)
- ease： 平滑过渡。等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0)
- ease-in： 由慢到快。等同于贝塞尔曲线(0.42, 0, 1.0, 1.0)
- ease-out： 由快到慢。等同于贝塞尔曲线(0, 0, 0.58, 1.0)
- ease-in-out： 由慢到快再到慢。等同于贝塞尔曲线(0.42, 0, 0.58, 1.0)
- step-start： 等同于 steps(1, start)
- step-end： 等同于 steps(1, end)
- steps([, [start | end]]?)： 接受两个参数的步进函数。第一个参数必须为正整数，指定函数的步数。第二个参数取值可以是start或end，指定每一步的值发生变化的时间点。第二个参数是可选的，默认值为end。
- cubic-bezier(, , ,)： 特定的贝塞尔曲线类型，4个数值需在[0, 1]区间内；

animation-delay:

检索或设置对象动画的延迟时间，默认值：0s；

- time：指定对象动画延迟的时间

animation-iteration-count

检索或设置对象动画的循环次数;

- infinite：无限循环
- number： 指定对象动画的具体循环次数

animation-direction:

检索或设置对象动画在循环中是否反向运动

- normal： 正常方向
- reverse： 反方向运行
- alternate： 动画先正常运行再反方向运行，并持续交替运行
- alternate-reverse： 动画先反运行再正方向运行，并持续交替运行

animation-play-state:

检索或设置对象动画的状态

- running： 运动
- paused： 暂停

animation-fill-mode:

检索或设置对象动画时间之外的状态

- none： 默认值。不设置对象动画之外的状态
- forwards： 设置对象状态为动画结束时的状态
- backwards：设置对象状态为动画开始时的状态
- both： 设置对象状态为动画结束或开始的状态

关键帧动画steps功能符深入介绍

https://www.zhangxinxu.com/wordpress/2018/06/css3-animation-steps-step-start-end/

cubic-bezier(): 贝塞尔曲线，则有linear，ease，ease-in，ease-out以及ease-in-out

steps()：逐步运动,简化出了step-start和step-end这两个关键字；

steps() 语法:

steps(number, position)

number数值。这个很好理解，表示把我们的动画分成了多少段；

position关键字。表示动画是从时间段的开头连续还是末尾连续。支持start和end两个关键字，默认是end

- start：表示直接开始。也就是时间才开始，就已经执行了一个距离段。于是，动画执行的5个分段点是下面这5个，起始点被忽略，因为时间一开始直接就到了第二个点：

- end：表示戛然而止。也就是时间一结束，当前距离位移就停止。于是，动画执行的5个分段点是下面这5个，结束点被忽略，因为等要执行结束点的时候已经没时间了：

steps()与填充模式animation-fill-mode

- animation-fill-mode有时候也会影响steps()的断点表现：animation: move 5s forwards steps(5, end);
- 动画只执行一次，因为没有设置infinite无限循环，而forwards虽然表示“前”，但同样和现实表现是反的，也就是动画结束时候元素保持动画关键帧最后的状态。于是，下面6个分段点都会执行，整个动画停止在第6个分段点上。

可以消减分段个数和动画运动的跨度，调整如下：

```
@keyframes move {
    0% { left: 0; }
    100% { left: 80px; }
}
```

也就是原来终点100px改成80px，同时CSS调用改成：animation: move 5s forwards steps(4, end);

也就是原来steps(5, end)改成steps(4, end)，最后100%这一帧交给forwards即可！

CSS面试真题

- 如何平移/放大一个元素
  - transform:translateX(100px)
  - tansform:scale(2)
- 如何实现3D效果
  - perspective:500px;
  - transform-style:preserve-3d;
  - transform: translate rotate ...
- CSS动画的实现方式有几种
  - transition
  - keyframes(animation)
- 过渡动画和关键帧动画的区别
  - 过渡动画需要有状态变化
  - 关键帧动画不需要状态变化
  - 关键帧动画能控制更精细
- 如何实现逐帧动画
  - 使用关键帧动画
  - 去掉补间(steps)
- CSS动画的性能
  - 性能不坏
  - 部分情况下优于JS

- 但JS可以做到更好
- 部分高危属性 box-shadow等