# html, body 默认高度 0

# DOM属性的操作

## setAttribute()

把指定属性设置或修改为指定的值

DOM.setAttribute(“class” , “box”); //DOM不是固定值

## removeAttribute()

删除指定属性

DOM.removeAttribute(“class”);

## getAttribute()

返回指定的属性值

var return = DOM.getAttribute(“class”);

在原属性值基础上添加新值：先获得旧值，再添加新值

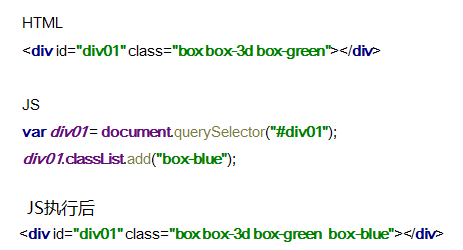
var result = DOM.getAttribute(“class”);

DOM.setAttribute(“class” , result + “ box1”) //box1前面加空格，class各个值之间有空格

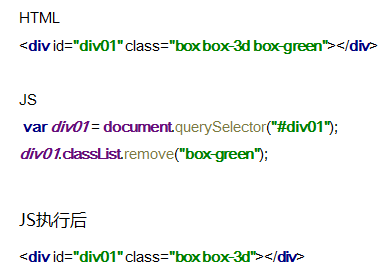
## classList

是HTML5为DOM提供的一个操作 class 属性的对象

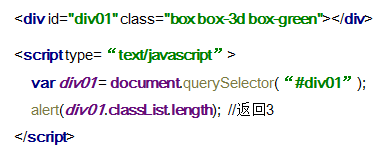
Dom.classList.add("box1"); //为一个DOM追加一个类属性值



Dom.classList.remove("box1"); //为一个DOM删除一个类



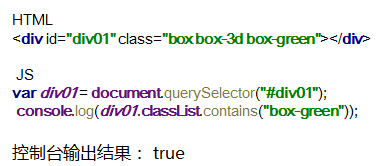
Dom.classList.length; //返回class 属性中 类选择器的个数



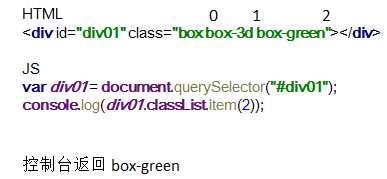
Dom.classList.toggle("box1"); //属性中有box1，删除，没有则增加



DOM.classList.contains("box1 "); //是否包含box1属性值



DOM.classList.item(2); //获取对应索引的属性值



# dataset 自定义属性

元素中以 data- 开头的属性，自定属性值提供程序使用，不需要给用户呈现。

**data-goods-desc 就是自定义属性，其语法规则要求，**

**1、必须以data-开头**

**2、单词之间建议使用 - 分割**

**3、不推荐使用大写字母**

获取自定义属性值：



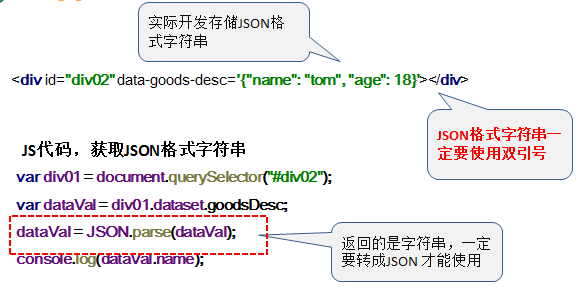
修改自定义属性值：



新增自定义属性及其值



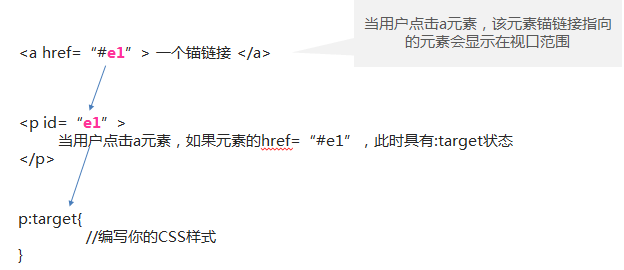
扩展用法：JSON格式字符串





# CSS3元素选择器

## :target 目标伪类选择器



## 结构化伪类选择器

### nth-of-type(参数)

* + E:nth-of-type(参数) E 表示CSS选择器。E选择器选中的元素**（必须为相同类型，同一层级的元素）**，根据后面的参数，再次“筛选”那个位置上的元素
  + 参数可选择值
    1. number 选择number指定位置上的元素
    2. odd 选择奇数位置上的元素
    3. even 选择偶数位置上的元素
    4. an+b 从第b个元素开始，每a个一组，选取第一个元素

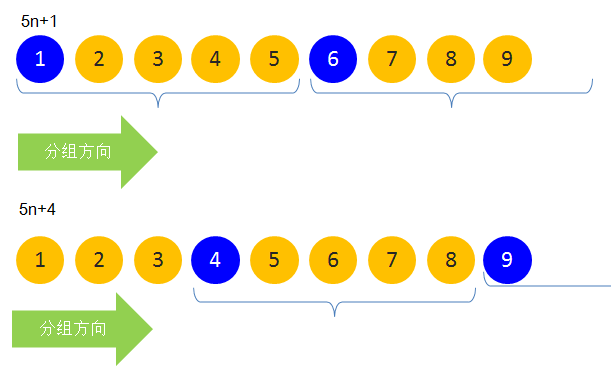
a

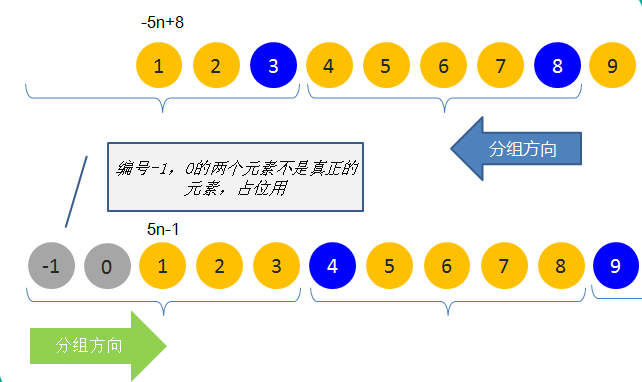
1.分组的方向（正数 从左向右，反之从右向左）

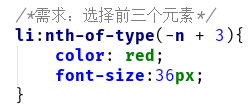
2.几个元素一组，当a为1时，一个一个选择

b

分组的起始位置（如果是负数，浏览器默认补齐；如果不写，相当于0，浏览器默认补齐）

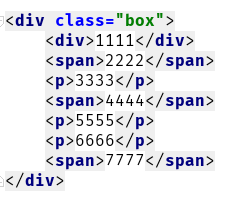
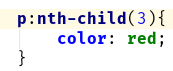




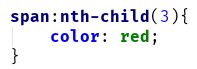
 

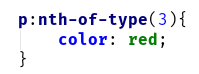
### nth-child(参数)

* + E:nth-child(参数) E 表示CSS选择器。E选择器选中的元素**（必须为相同类型，同一层级的元素）**，根据后面的参数，再次“筛选”那个位置上的元素
  + 参数可选择值
    1. number 选择number指定位置上的元素
    2. odd 选择奇数位置上的元素
    3. Even 选择奇数位置上的元素
    4. an+b 从第b个元素开始，每a个一组，选取第第一个元素

p:nth-child(3) /\*父元素的所有子元素中第三个，而且这个元素是p元素\*/

  /\*父元素中第三个元素不是span，样式没有生效\*/

  /\*父元素中所以p元素的第三个\*/

## :checked状态伪类

当check为checked时的伪类

Input check 配合 appearance 样式

### appearance

使用CSS3的appearance属性清除默认样式

-webkit-appearance: none; //-webkit-浏览器厂商

-moz-appearance: none; //-moz-浏览器厂商

appearance: none;

一个样式被添加厂商前缀

1.该样式正处于试验阶段。

2.浏览器特有的样式。

input[type=checkbox] {

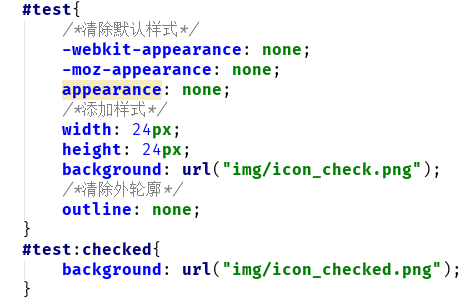
-webkit-appearance: none;

-moz-appearance: none;

appearance: none;

} //清除checkbox默认样式



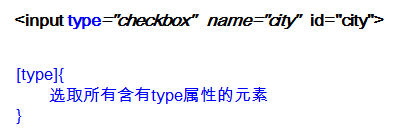


 未选中  选中  未清除外轮廓的选中

## 属性选择器

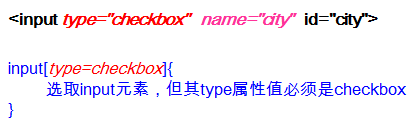
### [attribute]

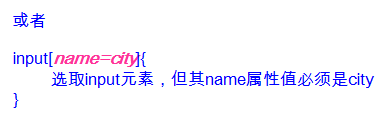
选择器用于选取带有指定属性的元素。



### [attribute=value]

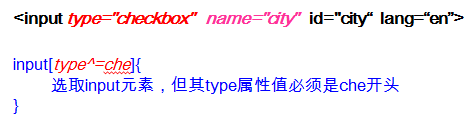
选择器用于选取带有指定属性和值的元素。





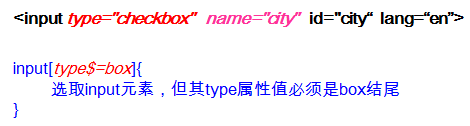
### [attribute^=value]

选择器匹配属性值以指定值开头的每个元素。



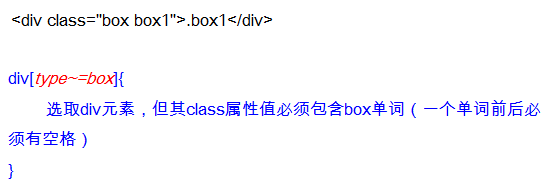
### [attribute$=value]

选择器匹配属性值以指定值结尾的每个元素。



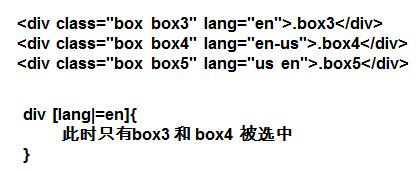
### [attribute~=value]

选择器用于选取属性值中包含指定词汇的元素。

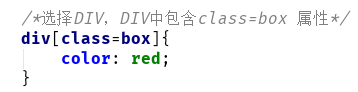


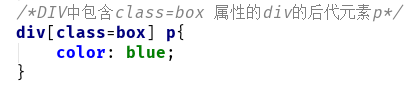
### [attribute|=value]

* 选择器用于选取带有以指定值开头的属性值的元素。
* 注意：该值必须是整个单词，比如 lang="en"，或者后面跟着连字符，比如 lang="en-us"。



### 搭配其他选择器使用

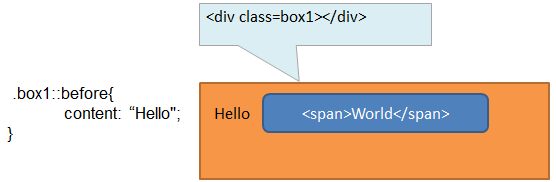




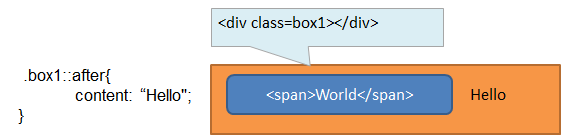
## 伪元素选择器

伪元素默认是行内元素

### ::befor



### ::after

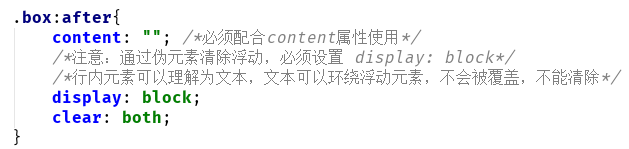


::after 与 ::before 注意事项

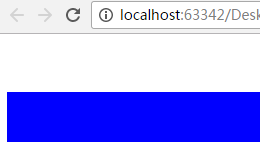
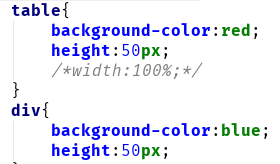
1. :: 是规范要求，实际开发中使用 : ，目的是兼容IE8
2. 必须配合content属性使用。
3. ::before ::after 相当于在元素内“动态”创建一个虚拟的inline元素，但该元素的文本内容无法被选中，该元素也无法通过JS获取其对应的DOM.

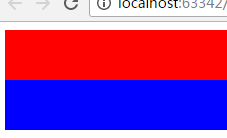
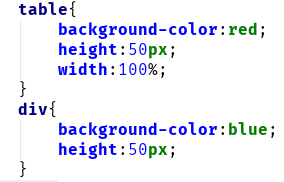
### 伪元素清除浮动

伪元素默认是行内元素，行内元素可以理解为文本，文本可以环绕浮动元素，不会被覆盖，不能清除浮动，注意：通过伪元素清除浮动，必须设置 display: block



但**更好的是设置display:table，**将伪元素设置为块元素只是为了清除浮动的影响，它里面其实是不需要内容的，也就是不需要宽高；因为同样是块元素，普通块元素不指定宽度时，默认是父元素的100%（浏览器在渲染页面时需要计算宽度为auto，不论是否有高度）；而table不指定宽度时是没有宽度的，浏览器就不需要计算了，少一次渲染，可以提高性能



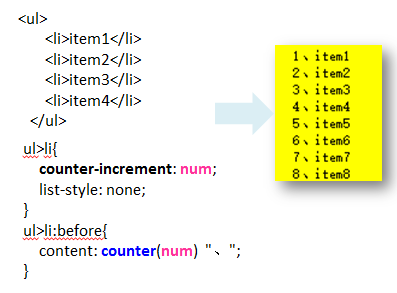


### CSS3计数函数

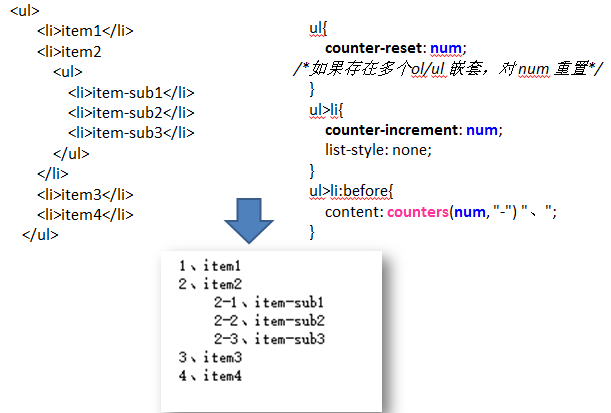
counter(counter-increment 属性值)

### ::befor配合counter(), counters()

::befor配合counter(num)



::befor配合counters(num,”str”)



1、为什么使用:?

兼容IE8

2、:before伪元素被添加的位置？

作为元素的第一个子节点

3、:after伪元素被添加的位置？

作为元素的最后一个子节点

4、content属性可选值？

- ""

- CSS的函数

- "" 和 函数混合输出

5、:before，:after伪元素默认元素类型？

-- inline

6、伪元素:before和:after与元素的区别？

1.伪元素无法被JS获取DOM

2.伪元素的内容无法被选中。

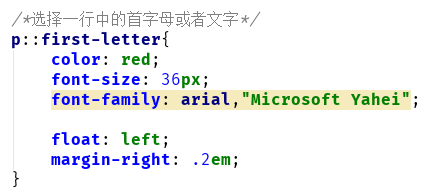
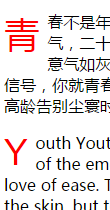
7、伪元素:before和:after与伪类的区别？

伪类- 表示元素处于某一状态时的样式，当状态消失，样式失效。

伪元素 是动态添加的一个元素。

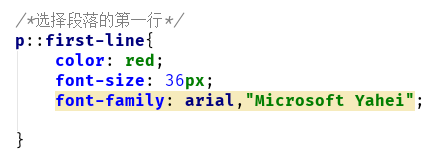
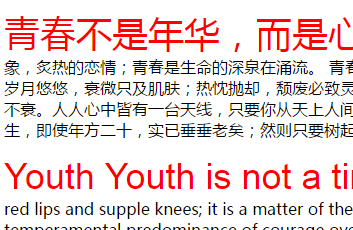
### ::first-letter

* 选中元素第一行文本的第一个字母或者汉字

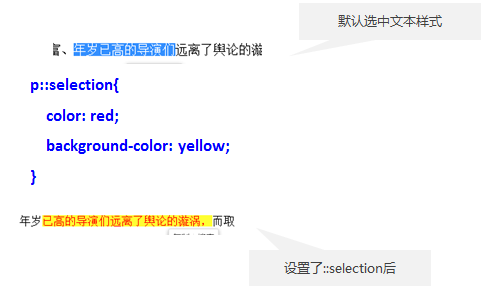
### ::first-line

* 选中元素第一行文本

### ::selection

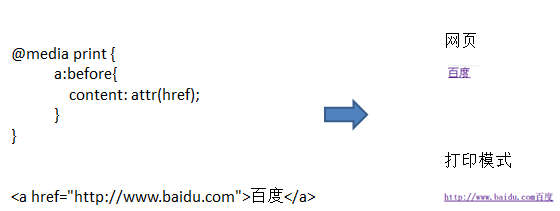
对选中文本样式设置，只能修改字体颜色、背景色



## 多媒体选择器

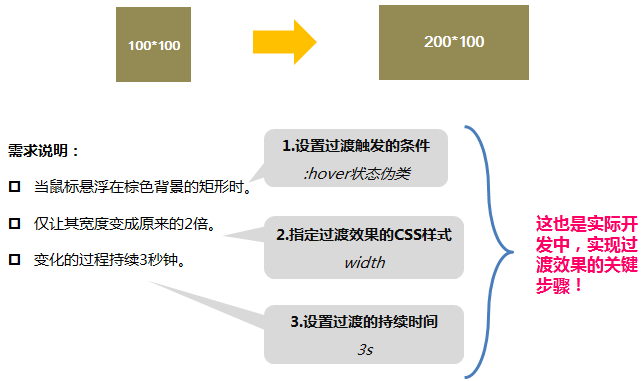
@media print 控制打印机页面样式

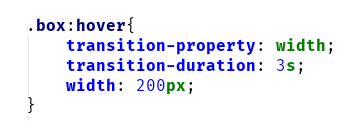
属性选择函数 attr(元素的属性名)，获取元素属性的属性值



# 过渡transition

过渡(transtion)是CSS3新增的一个样式属性。得益于CSS3提供过渡属性，让我们不再依赖JS或者flash来实现动画效果。





## 设置CSS样式属性具有过渡效果

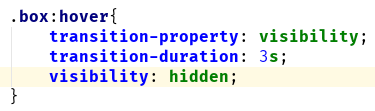
* 默认情况，浏览器会为每个元素设置,
  + transition-property: all
  + 参数all表示所有具有**中间值**的CSS样式属性名**，只要样式有值的变化，且设置了过渡时间**，就会发生过渡效果。
* 除非我们有以下需求，才需要特殊设置
  + **仅元素的宽度有过渡效果（此时即便有其他CSS样式属性值发生变化，也不发生过渡动画效果）**

transition-property: width

* + **仅宽度和字体颜色有过渡效果（注意：多个CSS样式使用逗号分隔）**

transition-property: width , color

* 中间值: CSS样式是数值或者可以转化成数值，颜色关键字可以转换为16进制
* 注意：类似于 visibility的CSS样式，无论如何也不会有过渡效果，**因为visibility的值既不是数值，也无法转为数值。但如果设置过渡时间了，在时间上过了过渡时间后，visibility会执行，相当于给visibility设置了延时调用**

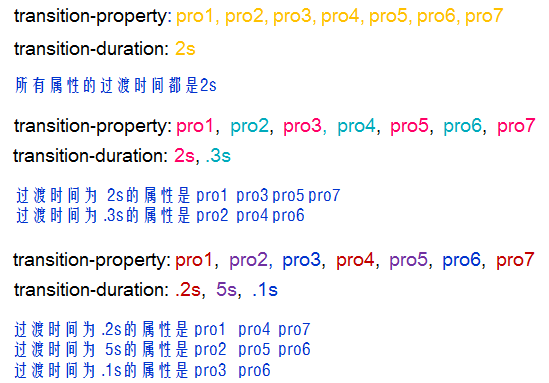


当鼠标在.box上悬停3秒后，.box消失

## 过渡时间

* transition-duration
  + **该属性是过渡的核心**，设置过渡效果持续的时间。单位秒(s)或者毫秒(ms)
  + **默认 0s ，即没有过渡效果（如果没有指定过渡时间，是无论如何也无法产生过渡效果的）**
* 过渡样式属性的可设置的值
  + transition-duration: .3s 过渡持续时间 300毫秒，推荐写法
  + transition-duration: 0.1s 过渡持续时间 100毫秒，**不推荐该写法**
  + transition-duration: 200ms 过渡持续时间 200毫秒
  + transition-duration: 2s 过渡持续时间 2秒
  + transition-duration: 2s, .5s **多个过渡时间使用逗号分隔， 通常配合控制多个过渡属性.**

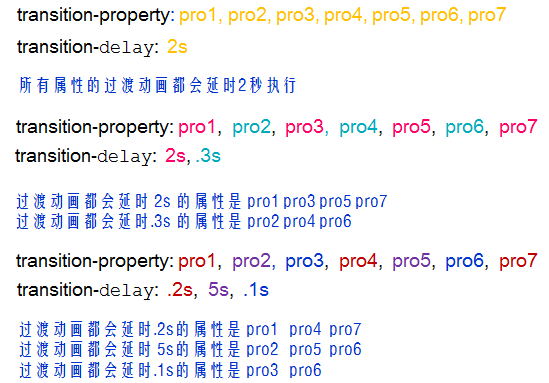
多个过渡时间，控制多个属性



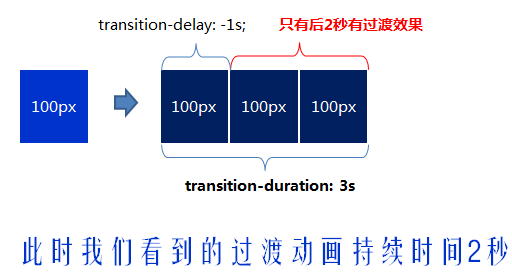
## 过渡延迟时间

* transition-delay
  + 设置过渡延迟执行时间。
  + 单位秒或者毫秒。
  + 默认 0s ，即立即执行过渡效果。**可以负值**。
* 过渡的参考值
  + transition-delay : .3s 过渡延迟300毫秒后执行，推荐写法
  + transition-delay : 0.1s 过渡延迟100毫秒后执行，**不推荐该写法**
  + transition-delay : 200ms 过渡延迟200毫秒后执行
  + transition-delay : 2s 过渡延迟2秒后执行
  + transition-delay : -1s **如果过渡持续时间3秒,实际看到的过渡效果只有后2秒( 3s – 1s )，前1秒会立即执行，并没有任何过渡效果。**
  + transition-delay : 2s, .5s 多个过渡时间使用逗号分隔， 通常配合控制多个过渡属性.

多个过渡延迟时间

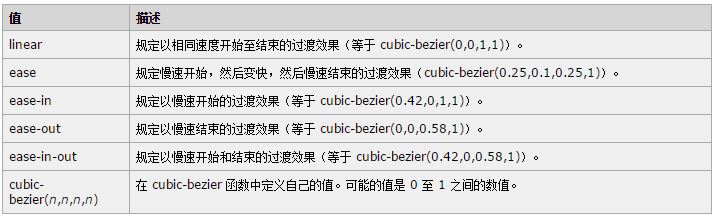


延迟过渡：负时间

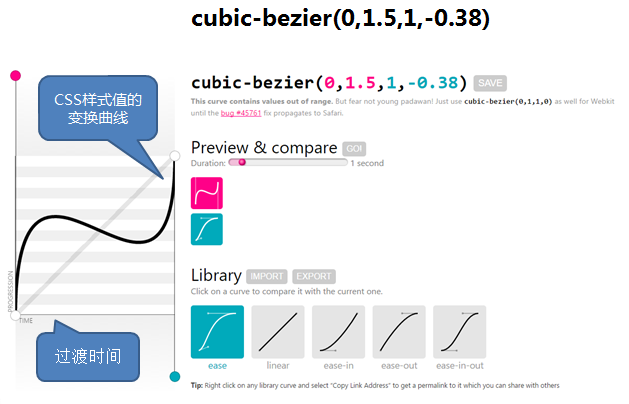


## 过渡效果

* **控制在整个过渡动画时间内不同时间段内变换的快慢，以及值的大小。CSS提供了5个关键字，默认**
  + transition-timing-function: ease



* 所有关键字都可以转换为cubic-bezier（），在实际开发中更倾向使用cubic-bezier（），我们可以通过在线工具生成该函数。



## 过渡简写

在实际开发过程中，我们推荐使用简写过渡。其语法如下，

* 1. 最简单的方式 ，只需要设置过渡时间。
     + transition: 过渡时间
  2. 标准简写方式。
     + transition: CSS属性名1 过渡时间 过渡效果 过渡延迟时间
  3. 多个属性控制，使用逗号分割简写方。
     + transition: CSS属性名1 过渡时间1 过渡效果1 过渡延迟时间1 , CSS属性名2 过渡时间2 过渡效果2 过渡延迟时间 2

特别说明

**当我们不显示指定CSS样式属性，浏览器默认为all，让所有样式属性的值是数值类型的CSS样式具有过渡效果。**

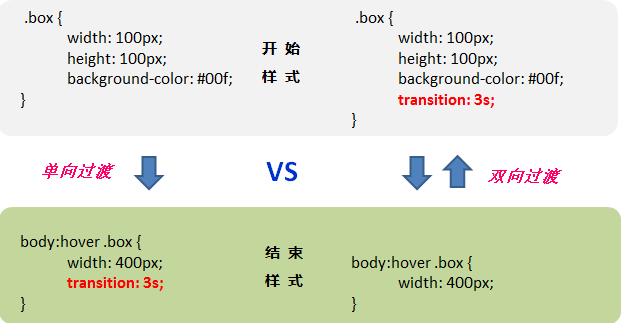
### 过渡简写覆盖问题 *默认: 逗号后的相同CSS名设置的过渡样式，会覆盖前面的过渡样式* transition:width 10s, all 2s -.5s;特殊情况*：*逗号后的相同CSS名设置的过渡样式，过渡时间小于等于零，使用前面的过渡样式。过渡时间为零的两种情况 1、 使用负延迟时间抵消 例如 transition:width 3s, all 1s -1s; 2、 显示设置0 例如 transition:width 3s, all 0s; 过渡时间小于零的情况

**使用负延迟时间 例如 transition:width 3s, all 1s -2s;**

## 单向过渡VS双向过渡

**当通过“状态伪类”触发过渡**

* + - 过渡样式设置在“样式开始值”位置（触发过渡，状态消失 都会产生过渡效果）
    - 过渡样式设置在“样式结束值”位置（只有触发过渡时，会产生过渡效果）



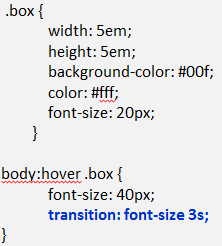
当在开始样式和结束样式都设置时间时，从开始到结束用结束的时间，从结束到开始 用开始的时间

## 触发过渡的三种方式

* 状态伪类
  + :hover
  + :active
  + :focus
* JS
  + 直接添加样式
* @media
  + 在不同的视口显示不同的样式

## 隐式过渡

* Firefox 与IE11 10会触发隐式过渡。



上述代码中，仅指定了字体大小的过渡，但是在firefox 和 IE11 10中导致了宽高的“隐式过渡”

## 过渡事件结束

* 过渡结束后，触发transitionend事件。

var element = document.getElementById("slidingMenu");

element.addEventListener("transitionend", function(event) {

element.innerHTML = “过渡时间结束!";

}, false);

## 过渡应用场景

* 缓慢缩放
  + 参考电商商品放大效果
* 缓慢位移
  + 参考移动端主菜单
* 淡入淡出
* 碰撞效果
* 旋转效果（配合 transform）