



CSS 布局那些事

讲师：许井龙

微信：ngsteel

2017年

一. 什么是布局?

二. 盒模型

1. 标准盒模型
2. 怪异盒模型
3. CSS属性box-sizing

三. 传统布局及相关技术

1. 右外边距失效
2. 负外边距 - 两列布局
3. BFC – 两列布局
4. 三列布局
5. 基于table-cell布局 (局部)

四. 伸缩盒模型 (flex)

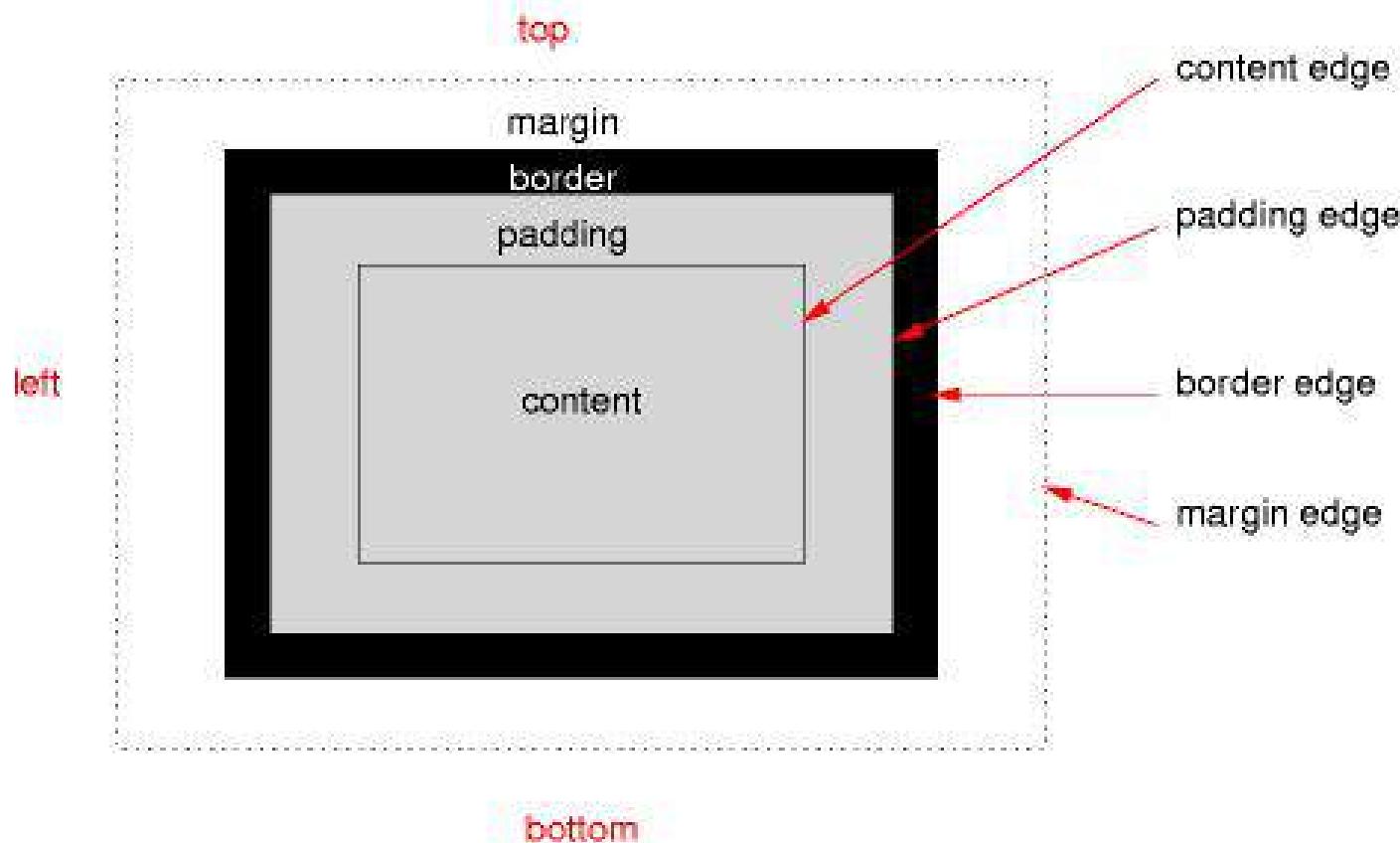
- 根据实际需求，把HTML元素通过CSS样式显示在网页中。

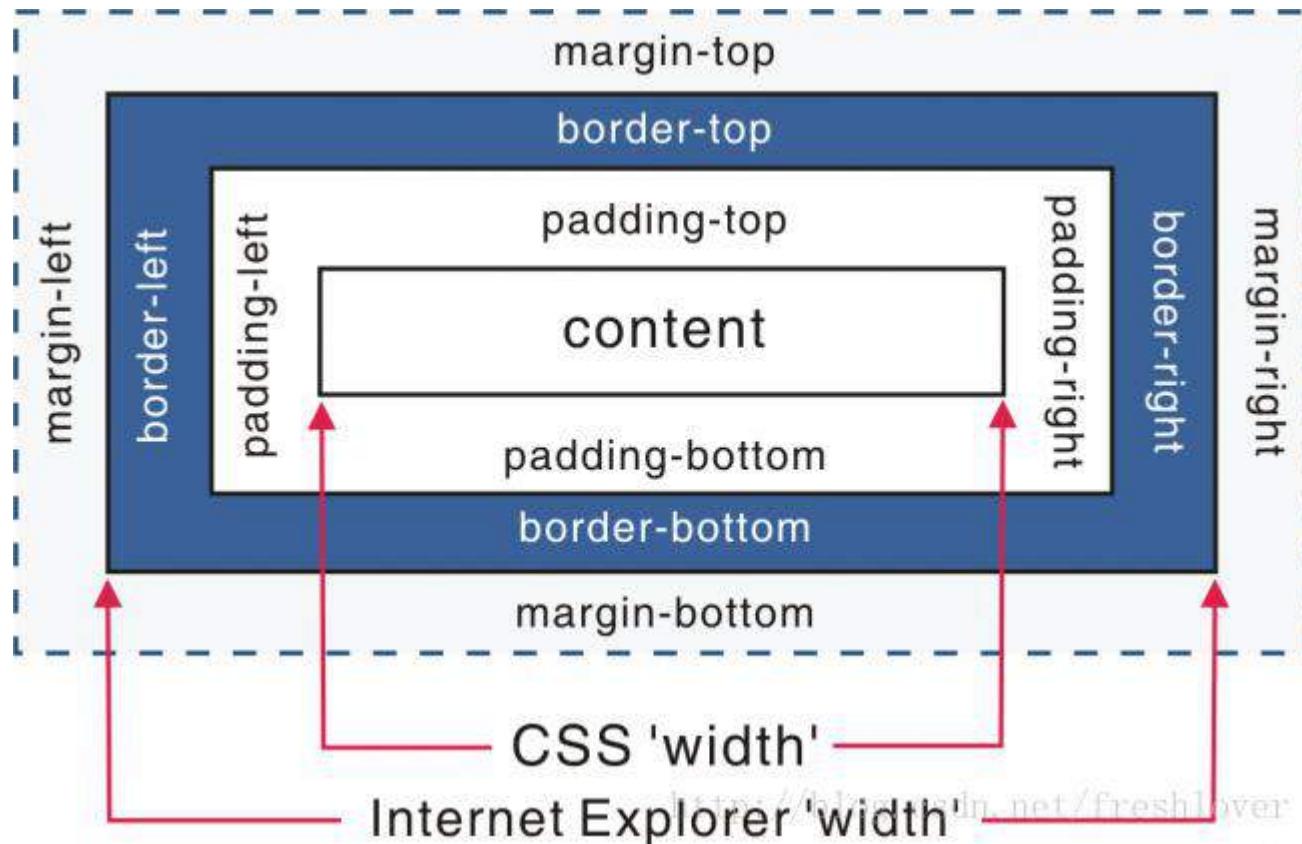
1. PC端布局特点

- 会为网页设置一个默认宽度，倾向像素单位（px）

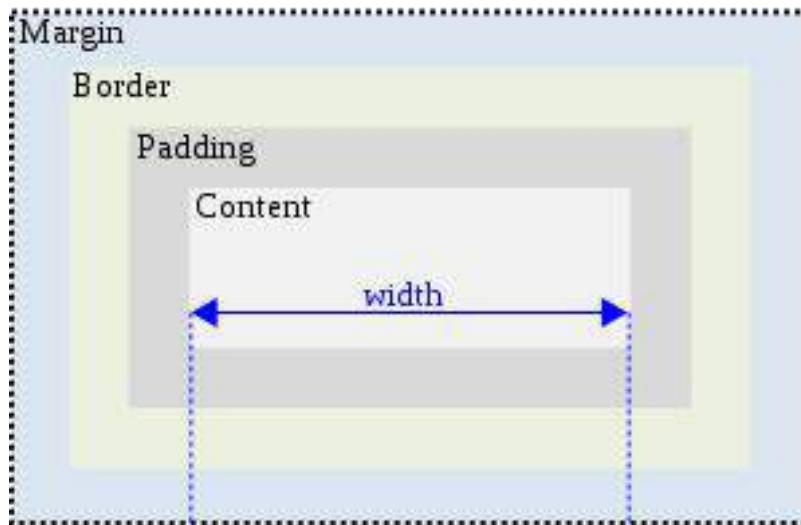
2. 移动端布局特点

- 基于视口作为网页宽度，更倾向使用相对单位（%，rem，vh，wh等）

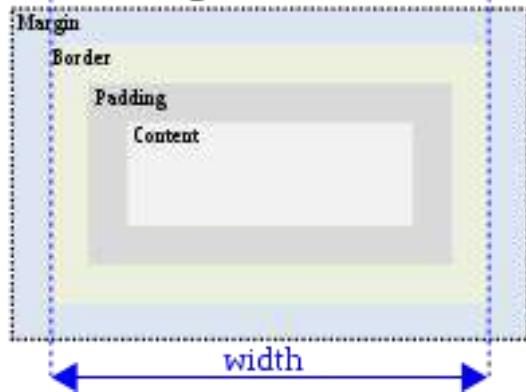




W3C box model



Internet Explorer box model



- 属性box-sizing
 - box-sizing: border-box; 元素设置宽度按 border-box 宽度计算
 - box-sizing: content-box; (默认值) , 元素设置宽度按 content-box 宽度计算

□ 父元素空间不足会导致子元素有外边距失效！

- 使用负外边会产生一个元素“悬浮”在另一个元素上面的效果。
- 注意：被覆盖的元素文本内容不会被覆盖！
- **使用场景：左侧自适应，右侧固定宽度布局效果**
 1. 为浮动元素（.box）设置右侧外边距为负
 2. 右侧外边距绝对值等于紧邻兄弟元素的宽度
 3. 为紧邻兄弟元素设置左浮动
- 4. 注意事项：.box需要添加一个子元素，子元素外边距为正值，该值等于父元素的负边距的绝对值。同时**不要给子元素指定宽度！！！**，通过上述处理，就不会出现**覆盖内容**的问题！

- 设置元素浮动，紧邻兄弟元素“占有”该元素位置，通过开启紧邻兄弟元素BFC（overflow: hidden），防止其上方出现浮动元素。
- **使用场景：左侧固定，右侧自适应**

- 综合负右外边距，BFC方式，实现中间自适应，两侧固定布局。

- 为元素设置 `display: table-cell` 该元素就具有了`<td>`的特性
 - 可以通过为该院设置`vertical-align` 轻松实现对齐内容及其子元素在垂直方向的位置
 - **我们推荐在局部布局中使用！不推荐整体布局！**

制作多列等宽自适应布局

- 1.父元素 `display: table; width: 100%`
- 2.布局元素（子元素）`display: table-cell;`
- 3.子元素之间的空隙，通过一个正常的

分割即可。
- 4.如果存在多行，需要在包裹一个 `display: table-row` (`<tr>`)





CSS2 定位 复习

讲师：许井龙

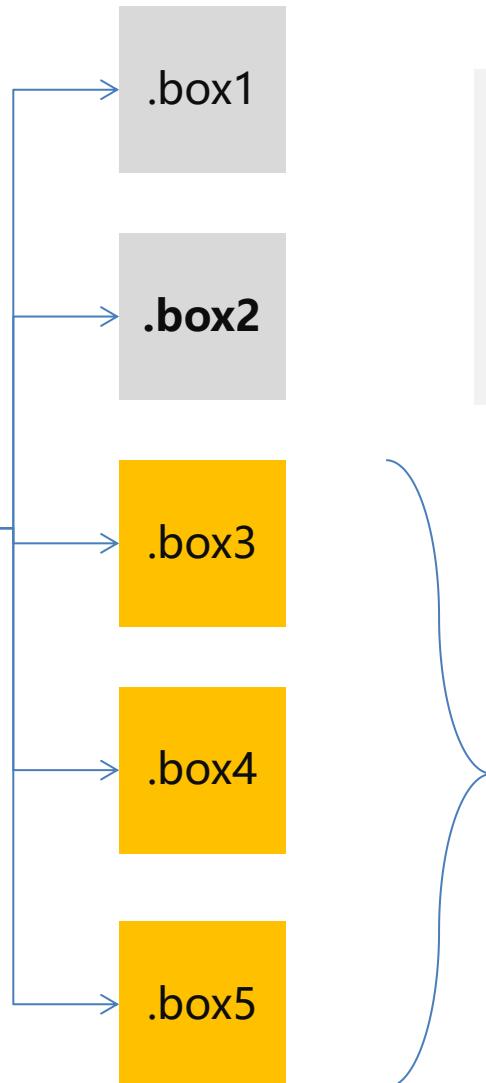
ngsteel@qq.com



- 兄弟元素
- 浮动
- 相对定位
- 绝对定位
- 固定定位

DOM树

父元素



```
<div class="box">
  <div class="box1"></div>
  <div class="box2"></div>
  <div class="box3"></div>
  <div class="box4"></div>
  <div class="box5"></div>
</div>
```

.box3, .box4, .box5
是.box2的兄弟元素

浮动

左浮动 float: left	清除左浮动 clear: left
右浮动 float: right	清除右浮动 clear: right
清除浮动	可为浮动元素的 兄弟元素 设置clear样式清除浮动，最简单的方式 clear: both
如果父元素未设置高度，子元素的浮动会造成父元素高度塌陷。	<p>为父元素设置 overflow: hidden。防止高度塌陷。</p> <p>注意！！！上述方案有“潜在风险”！！！</p> <p>如果浮动元素的内容溢出，会被“隐藏掉”。</p> <p>理想解决方案：通过为父元素添加一个伪元素，并未其设置清除浮动解决。例如，</p> <pre>.box:after{ content: ""; display: table; clear: both; }</pre>

定位对比

对比项	相对定位	绝对定位	固定定位
定位“参照物”	相对元素自身位置	相对于离其最近且设置了 position: relative/absolute 的祖先元素，否则相对于初始化块（可以理解为相对于视口） 前提：必须设置top、right、bottom、left，否则上述结论不成立	相对于“初始化块”，可以理解为“视口”。 前提：必须设置top、right、bottom、left 前提：必须设置top、right、bottom、left，否则上述结论不成立
元素变化	升级为块元素，如果未设置宽度，宽度“清零”。相当于所有的auto值失效		
层级	配合 z-index 设置元素层级，如果设置层级相同，兄弟元素的层级更高。		
元素位移	配合top、right、bottom、left 对齐进行位置移动，也可以使用margin，但是我们不推荐使用margin。		
是否脱离文档流	不脱离	脱离	
如果父元素未指定高度，是否会造成父元素高度塌陷？	不塌陷	塌陷 注意：定位造成的父元素高度塌陷 不可以 通过设置父元素 overflow: hidden 方式解决！！！	

定位对比

对比项	相对定位	绝对定位	固定定位
宽度 100%	相对元素的父元素宽度	相对于离其最近且设置了 position: relative/absolute 的祖先元素，否则相对于初始化块宽度，也可以理解为相对于视口的宽度。	相对于初始化块宽度，也可以理解为相对于视口的宽度。





CSS2 背景

讲师：许井龙

ngsteel@qq.com

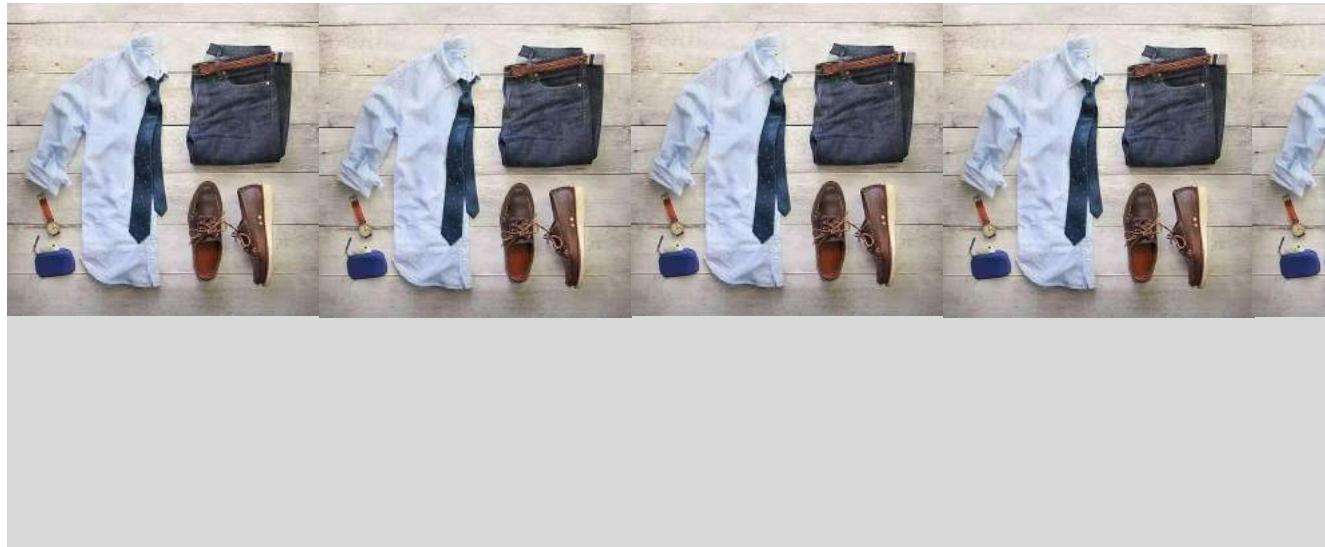
- 元素背景色
- 元素背景图像
- 元素背景图像平铺
- 元素背景图像位置
- 元素背景图像是否固定或者随着页面的其余部分滚动。
- 元素背景简写

- 设置元素背景色 **background-color**, 其可选值
 - **color_name** 规定颜色值为颜色名称的背景颜色 (比如 **red**) 。
 - **hex_number** 规定颜色值为十六进制值的背景颜色 (比如 **#ff0000**) 。
 - **rgb_number** 规定颜色值为 **rgb** 代码的背景颜色 (比如 **rgb(255,0,0)**) 。
 - **rgba_number** 规定颜色值为 **rgba** 代码的背景颜色 (比如 **rgb(255,0,0, .5)**) 。表示**50%**透明的红色。
 - **transparent** 默认。背景颜色为透明。等同于 **rgb(0,0,0, 0)**
 - **inherit** 规定应该从父元素继承 **background-color** 属性的设置。

- 背景图引入， **background-image** 可选值，
 - url('URL') 指向图像的路径。
 - none 默认值。不显示背景图像。
 - inherit 规定应该从父元素继承 background-image 属性的设置。

- 背景图平铺 background-position 可选值，
 - repeat 默认。背景图像将在垂直方向和水平方向重复。
 - repeat-x 背景图像将在水平方向重复。
 - repeat-y 背景图像将在垂直方向重复。
 - no-repeat 背景图像将仅显示一次。
 - inherit 规定应该从父元素继承 background-repeat 属性的设置。



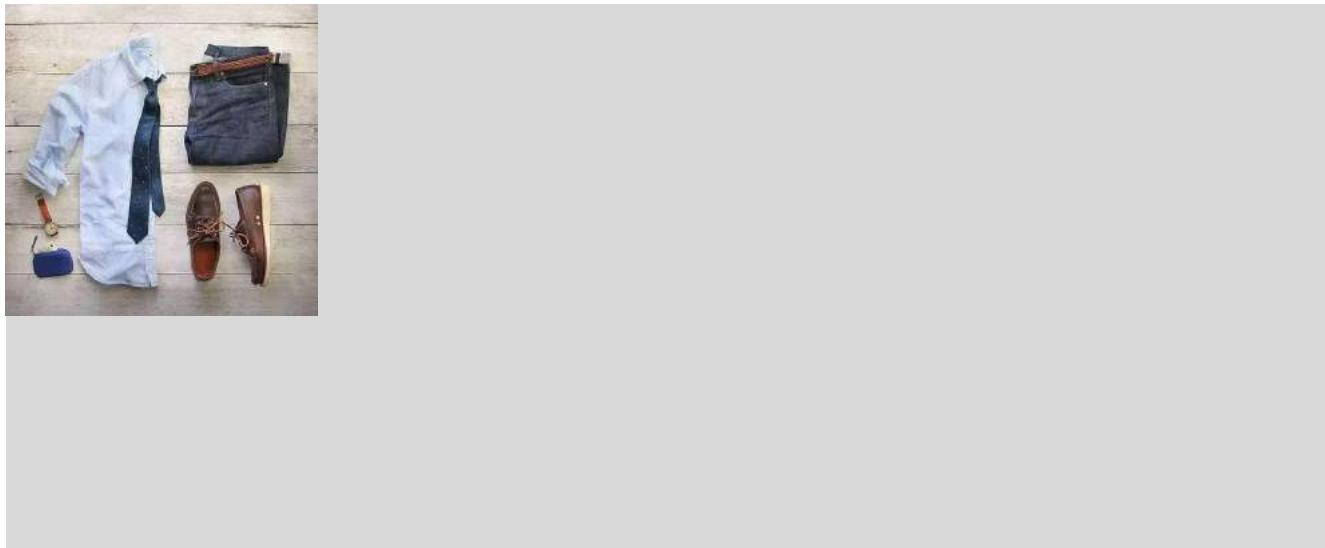






元素大小 500px * 300px
背景图大小 200px * 200px

- 背景图位置 background-position 默认值
 - 0% 0%。
 - 或者 left top
 - 或者 0 0



生效前提：

图片足够小，元素足够大

background-position: no-repeat

- 背景图位置 background-position
 - 50% 50%。
 - Center
 - 150px 50px

元素大小 500px * 300px
背景图大小 200px * 200px



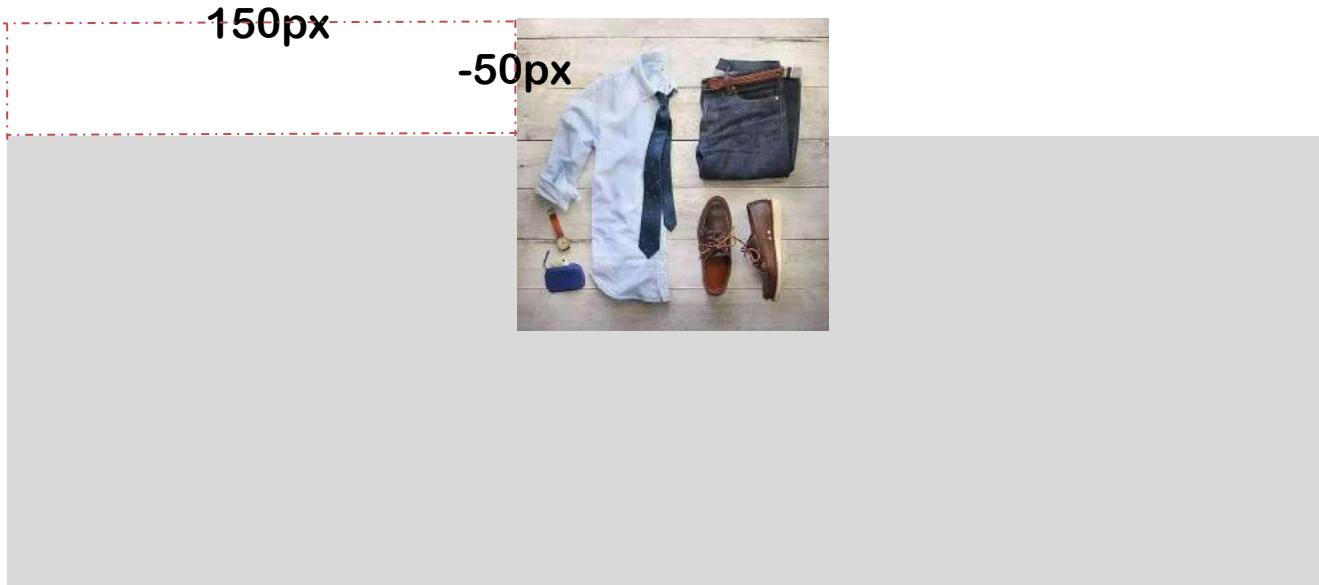
生效前提：

图片足够小，元素足够大

background-position: no-repeat

元素大小 500px * 300px
背景图大小 200px * 200px

- 背景图位置 background-position
 - 50% -50%。
 - 150px -50px



生效前提：

图片足够小，元素足够大

background-position: no-repeat

移动背景图显示不同的图标

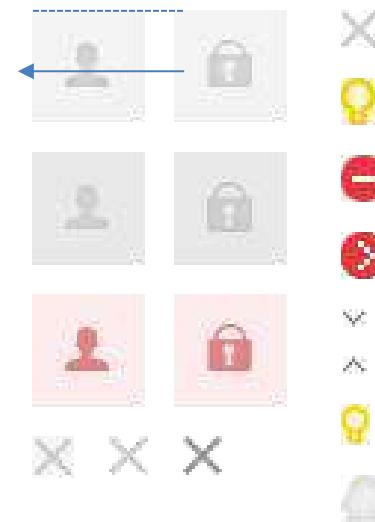
空隙10px

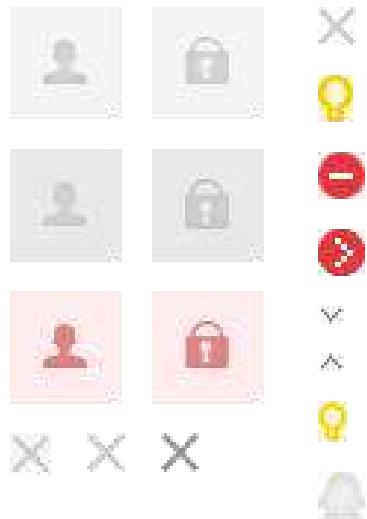
图宽38px

向左 **-48px**

左上角**6**个图标大小**38px * 38px** 如果显示锁头的图标

```
.box{  
    width: 38px;  
    height: 38px;  
    background-position: -48px 0;  
    background-image: url(pwd-icons-new.png);  
    background-repeat: no-repeat;  
}
```





CSS雪碧 即[CSS Sprite](#)，也有人叫它CSS精灵，是一种CSS图像合并技术，该方法是将小图标和背景图像合并到一张图片上，然后**利用css的背景定位来显示需要显示的图片部分。**

优点：减少加载网页图片时对服务器的请求次数

1. 提高页面的加载速度
2. 减少鼠标滑过的一些bug

缺点：

1. 雪碧图制作麻烦
2. 网页中雪碧图呈现的图象无法被打印
3. 用CSS编写比较麻烦

- 商品图片，网站Logo等重要的图片，我们不建议使用雪碧图。由于这些图片传达重要的信息的。把其放在雪碧图中会导致一个网站缺乏可访问性的，也会降低 HTML 中 title 和 alt 的潜在益处。

- **background-attachment** 可选值，
 - scroll 默认值。背景图像会随着页面其余部分的滚动而移动。
 - fixed 当页面的其余部分滚动时，背景图像不会移动。
 - inherit 规定应该从父元素继承 **background-attachment** 属性的设置。

```
body{  
    background-position: center;  
    background-image: url(bg.jpg);  
    background-repeat: no-repeat;  
    background-attachment: fixed;  
    height: 2000px;  
}
```



- background: 颜色 背景图 是否平铺 是否随页面滚动 背景图位置
- background: #00FF00 url(bgimage.gif) no-repeat fixed top;





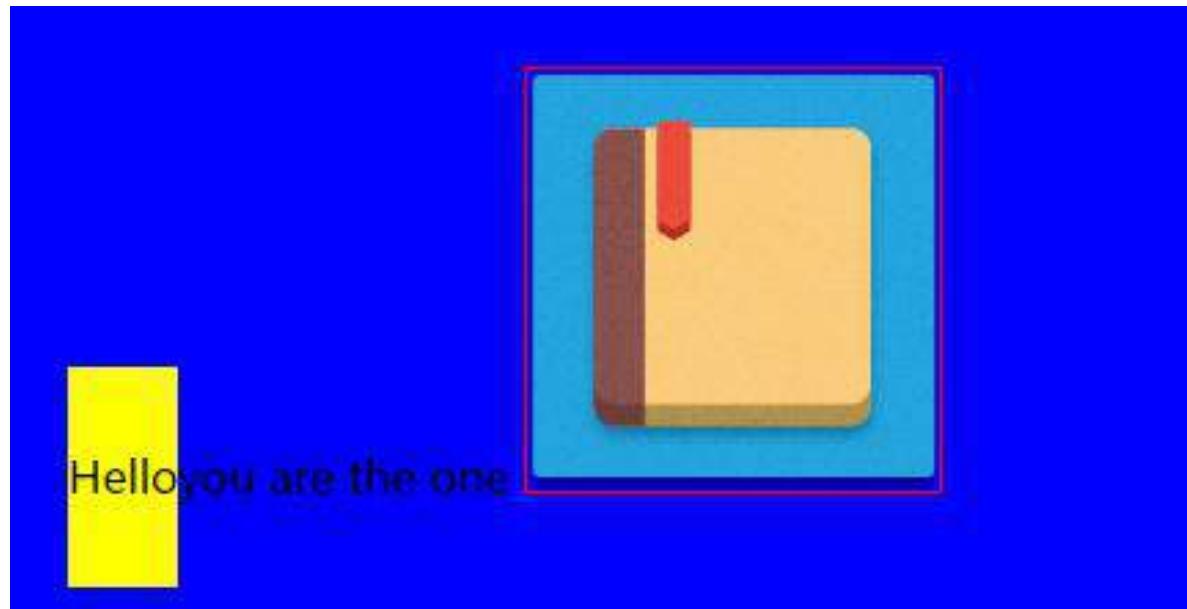
垂直对齐 vertical-align

讲师：许井龙

微信：ngsteel (承神之佑)

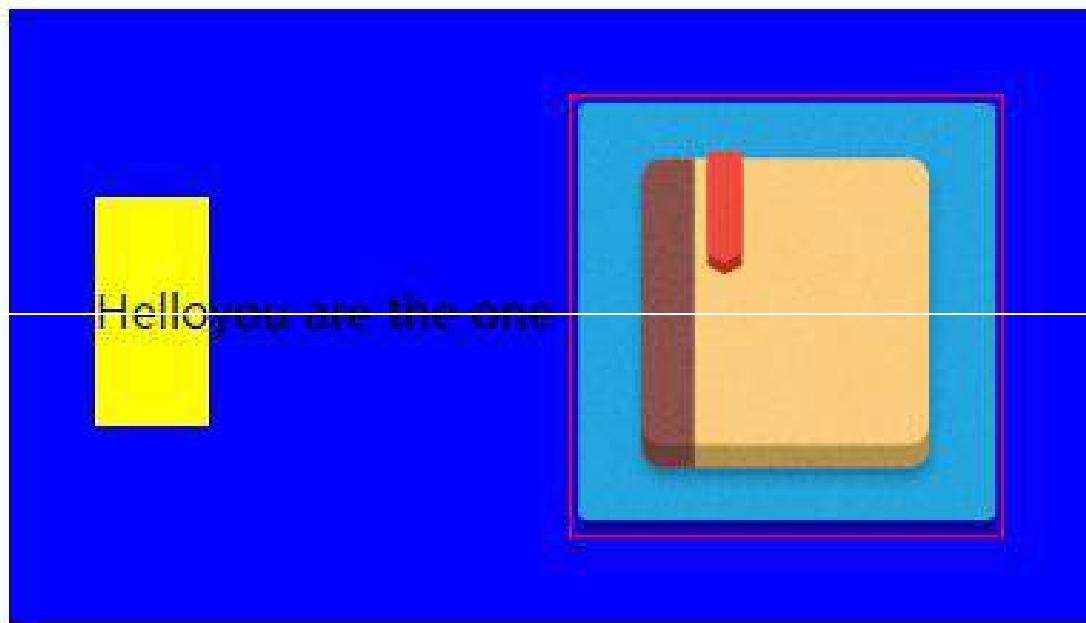


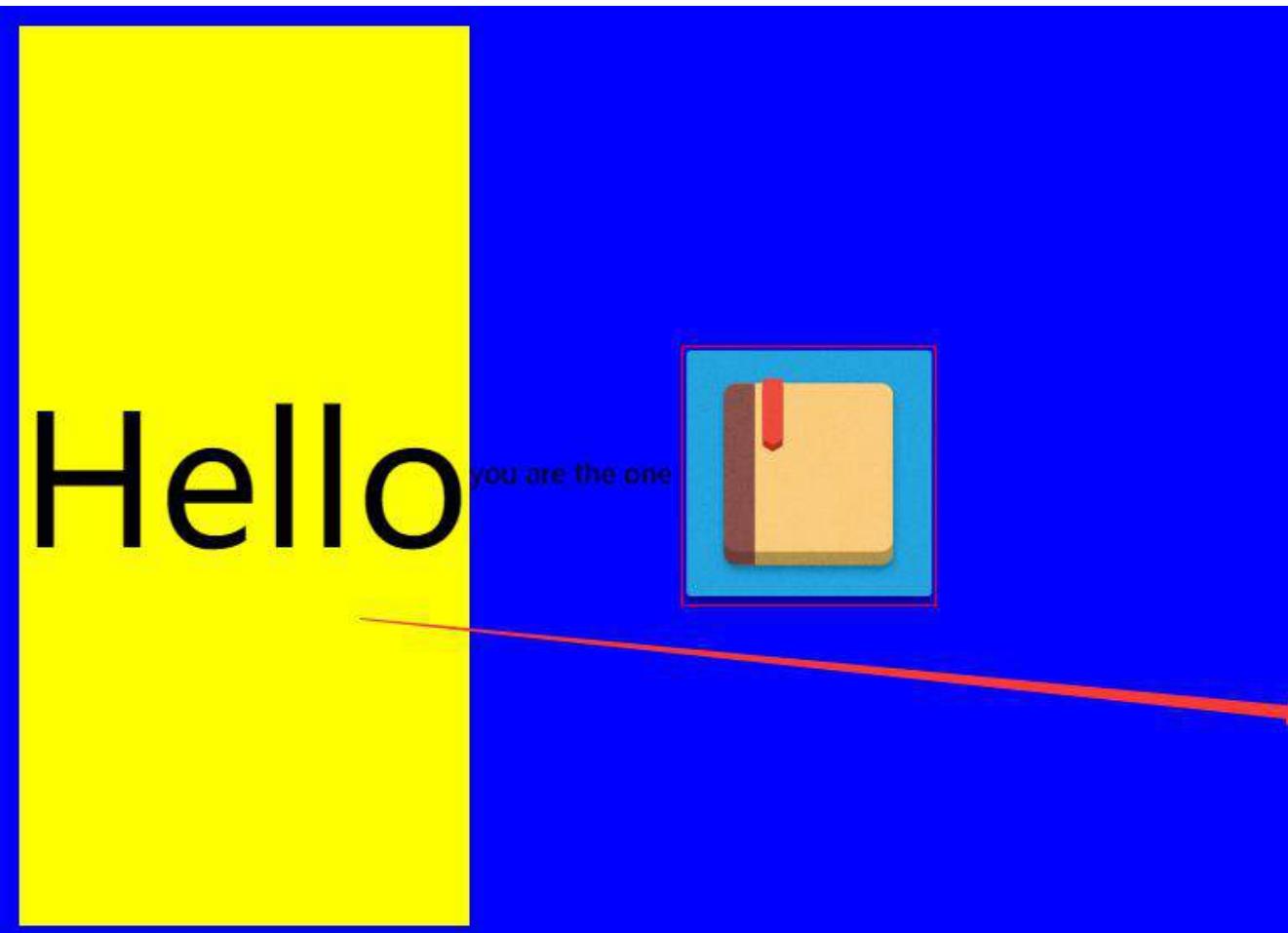
1. 一行内 (line box) , 文字和替换内容的对齐的基准线
2. 基线位于一行中父元素小写字母 x 的底部



黄色块: inline-block

- vertical-align: middle
 - 同一行父元素小写字母x的 $\frac{1}{2}$ 处
 - 该属性只在inline, inline-block, table-cell 生效。





```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <!--<div class="box box1">Hello</div>-->
    <div class="box2">
      <span>Hello</span> == $0
      "you are the one"
      <br>-->
      <i>C</i>-->
      
      <!--Hello<span>World</span>-->
    </div>
  </body>
</html>
```

Styles Event Listeners DOM Breakpoints Properties

Filter :hover .cls +

```
element.style { }
```

```
span { line-height: 5; background-color: yellow; display: inline-block; font-size: 107px; vertical-align: middle; }
```

Console

- 例如，`top`, `bottom`等了解即可。

- <td>
- display: table-cell





hasLayout

讲师: 许井龙
ngsteel@qq.com

- 简介
- 开启或关闭 hasLayout

- hasLayout可以简单看作是IE5.5/6/7中的BFC(**Block Formatting Context**)。

回顾

BFC就是一个块级元素要么自己对自身内容进行组织和尺寸计算，要么由其containing block来组织和尺寸计算。

- 当hasLayout为true时(就是所谓的“拥有布局”), 相当于元素产生新BFC, **块级元素自己对自身内容进行组织和尺寸计算**;
- 当hasLayout为false时(就是所谓的"不拥有布局"), 相当于元素不产生新BFC, **元素由其所属的containing block进行组织和尺寸计算**
- 和产生新BFC的特性一样, hasLayout无法通过CSS属性直接设置, 而是通过某些CSS属性间接开启这一特性。不同的是**某些CSS属性是以不可逆方式间接开启**。

- object.currentStyle.hasLayout

- display: inline-block
- height: (除 auto 外任何值)
- width: (除 auto 外任何值)
- float: (left 或 right)
- position: absolute
- writing-mode: tb-rl
- zoom: (除 normal 外任意值)

- IE7 还有一些额外的属性(不完全列表)可以触发 hasLayout：
 - min-height: (任意值)
 - min-width: (任意值)
 - max-height: (除 none 外任意值)
 - max-width: (除 none 外任意值)
 - overflow: (除 visible 外任意值, 仅用于块级元素)
 - overflow-x: (除 visible 外任意值, 仅用于块级元素)
 - overflow-y: (除 visible 外任意值, 仅用于块级元素)
 - position: fixed

- 虽然我现在已经不用再适配IE5.5/6/7了，但理解 hasLayout还是很有必要的。也算是可以理解为从另一个角度学习BFC吧！



- 视觉格式化模型的一个重要概念
- 可以把它理解为矩形，为其内部的元素提供一个参考，元素的尺寸，位置往往有该元素所在包含块决定。所有有时我们也把包含块称为“定位参考框”或者定位坐标参考系
-
- 一个元素的的**位置**和**大小**有时是通过相对于一个特定矩形盒子（containing block）计算，一个元素的包含块定义
 - 浏览器（user agent）选择根元素(html)作为包含块，也称之为初始包含块
 - 其他元素，除非元素使用绝对位置，包含块有最近的块级祖先元素盒子的内容边界组成。
 - position: fixed 包含块由视口建立
 - position: absolute 包含块由最近的 position 不是 static 的祖先建立，



包含块及初始包含块

containing block

讲师: 许井龙
ngsteel@qq.com



- 包含块：视觉格式化模型的一个重要概念
- 可以把包含块理解为矩形，为其内部的元素提供一个参考，元素的尺寸，位置往往有该元素所在包含块决定。所有有时我们也把包含块称为“定位参考框”或者定位坐标参考系

- 一个元素的的位置和大小有时是通过相对于一个特定矩形盒子 (containing block) 计算。
- 一个元素的包含块定义如下，
 - 浏览器 (user agent – 用户代理) 选择根元素(html)作为包含块，也称之为初始包含块。
 - 正常文档流以及浮动的元素，包含块由最近的块级祖先元素 (父元素) 盒子的 内容边界 (content-box) 组成。
 - position: absolute，包含块由最近的 position 不是 static (absolute / relative) 的祖先建立，默认是基于初始包含块。
 - position: fixed，包含块由视口建立

1. **正常文档流元素** (position: static) 以及**浮动元素**, 其包含块是父元素的content-box.
2. **绝对定位元素** (position: absolute) 其包含块是离其最近且开启了定位的**祖先元素的padding-box**。如果祖先元素没有开启定位, 其包含块是初始包含块 (初始包含块与视口等大, 可以理解为html元素的 margin-box)
3. **固定定位元素** (position: fixed) 其包含块是视口, 元素会随着视口位置和大小的改变而改变。

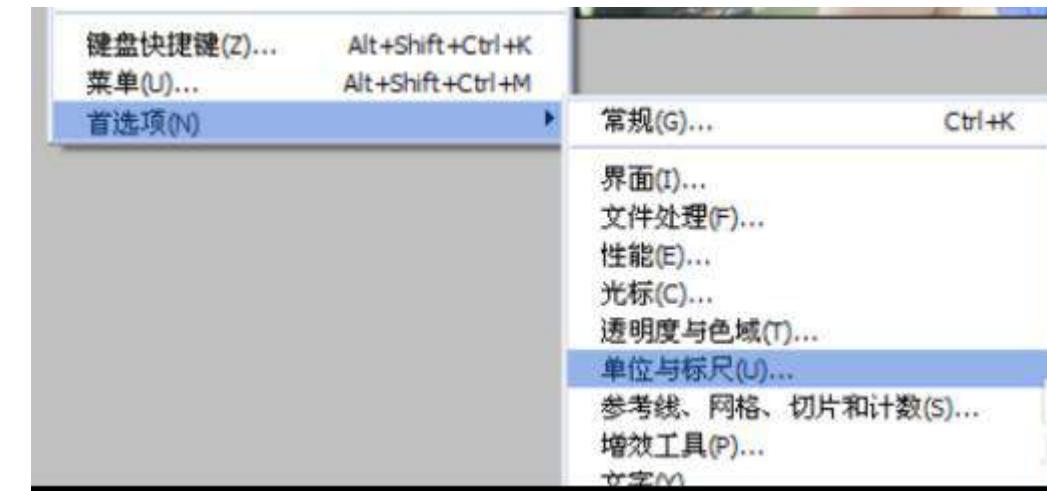
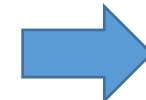


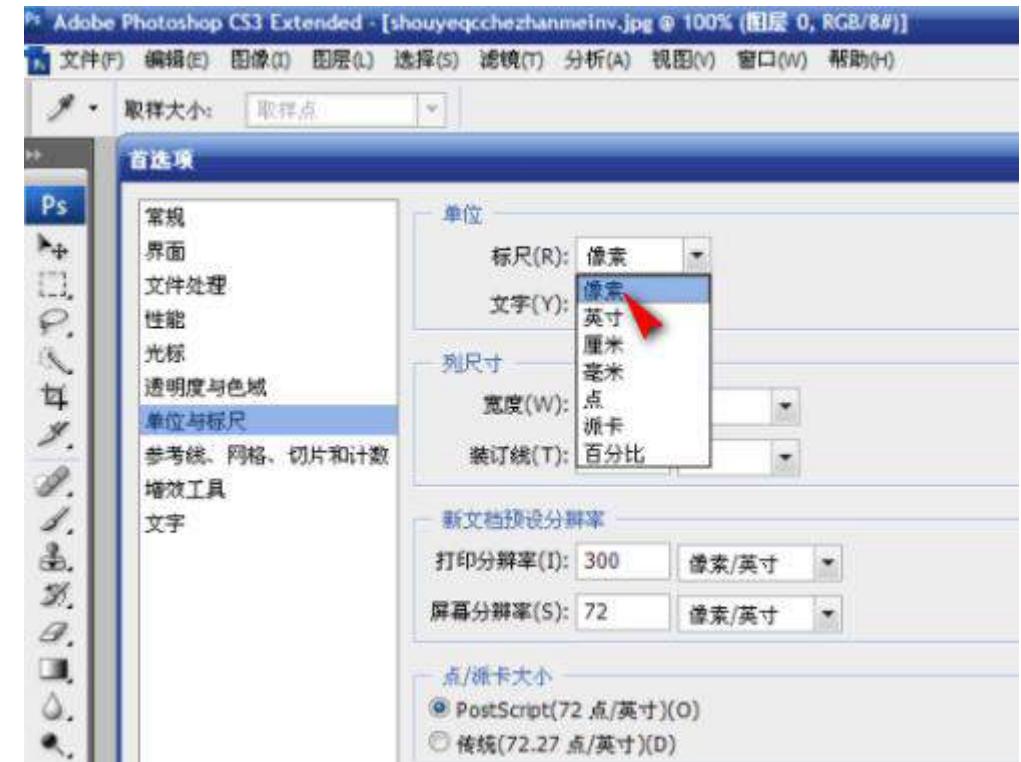
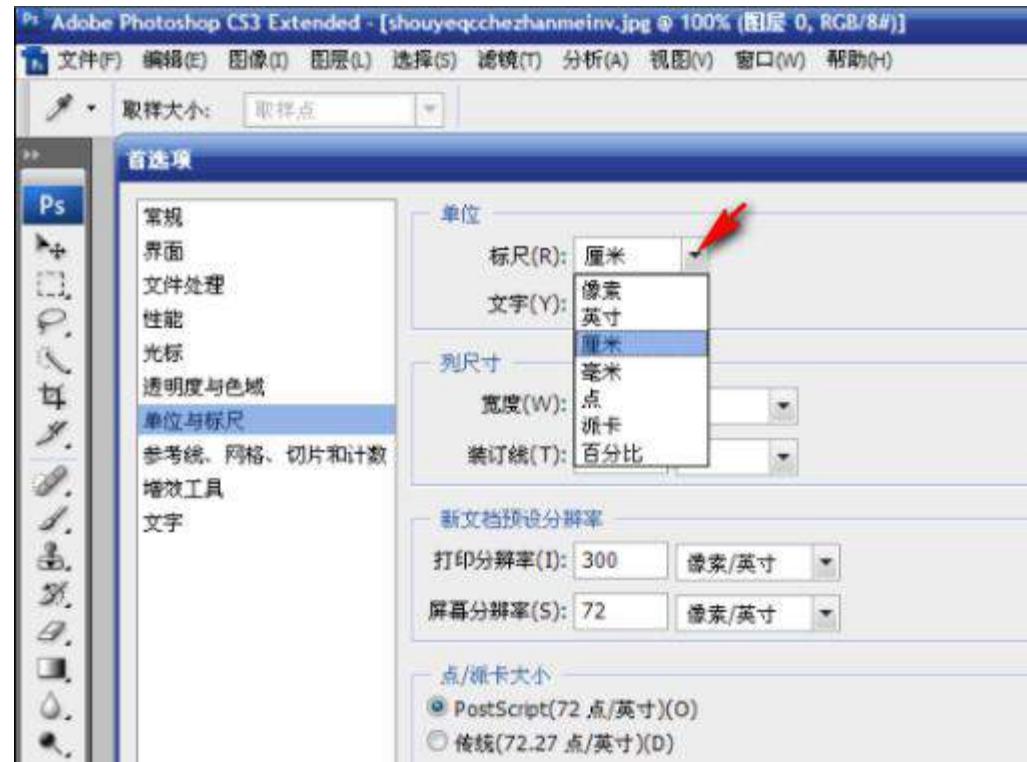
PS简单操作

1、设置PS默认的单位与标尺

- 编辑->首选项->单位与标尺

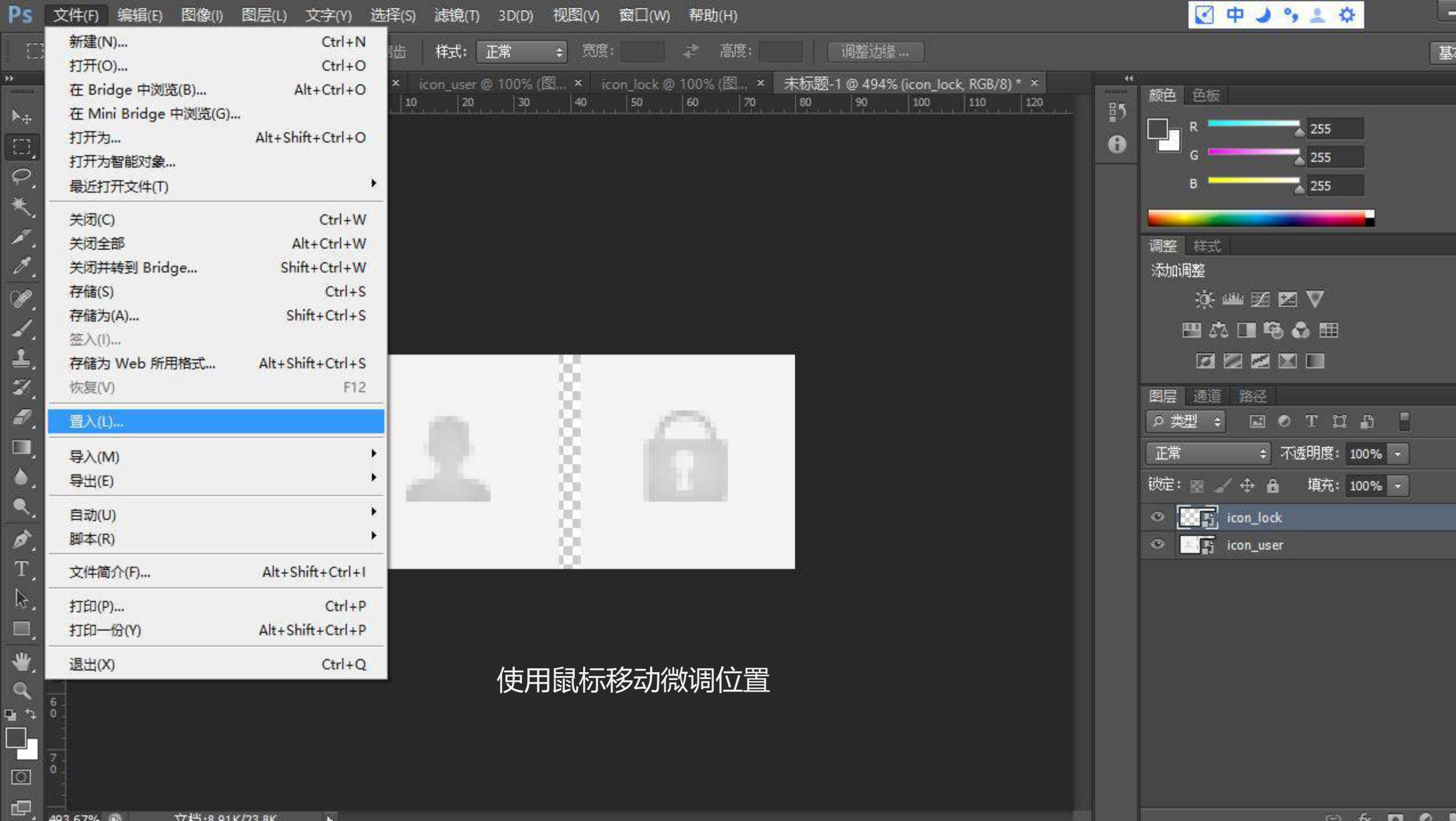
修改PS的默认单位





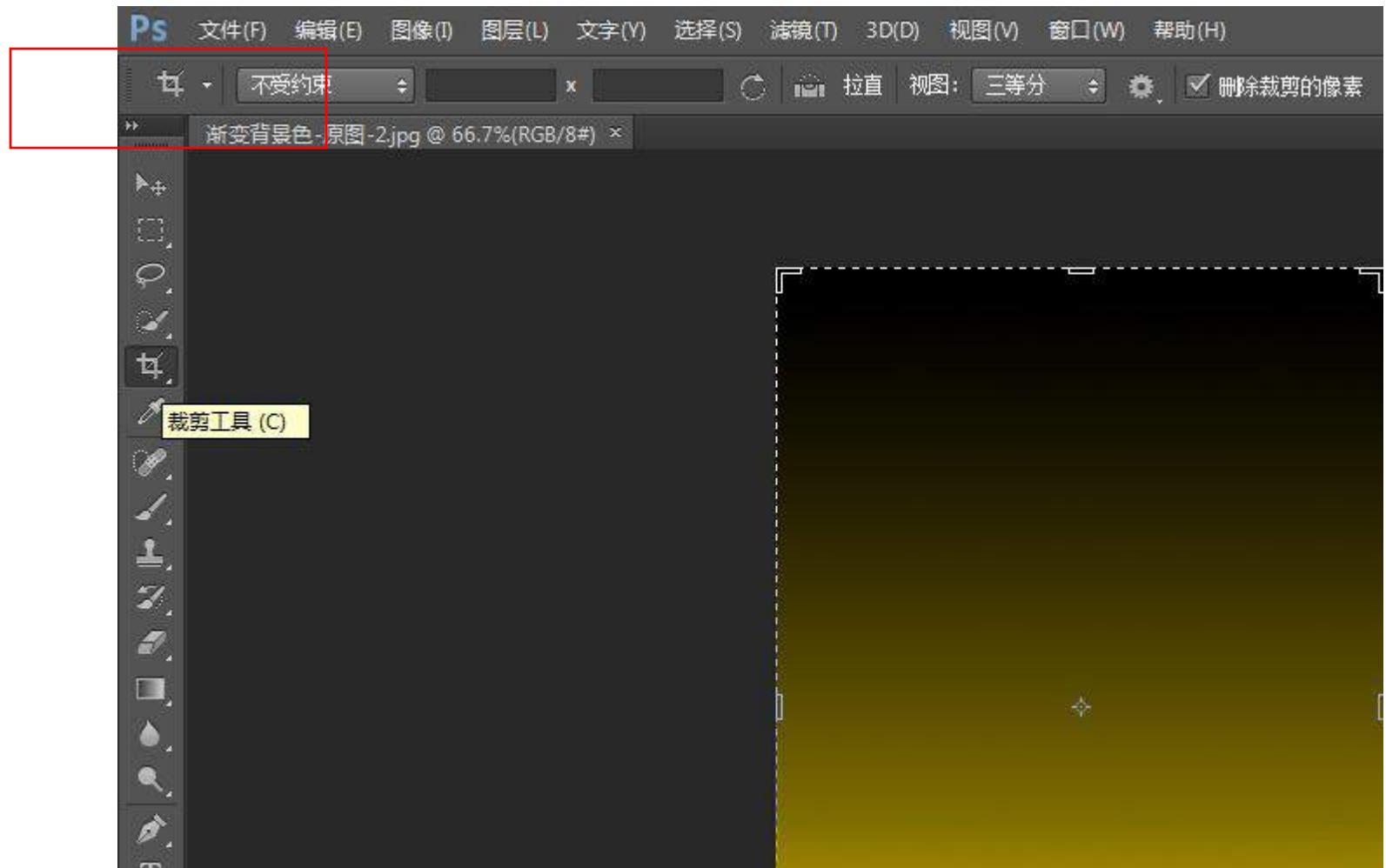
2、合成精灵图

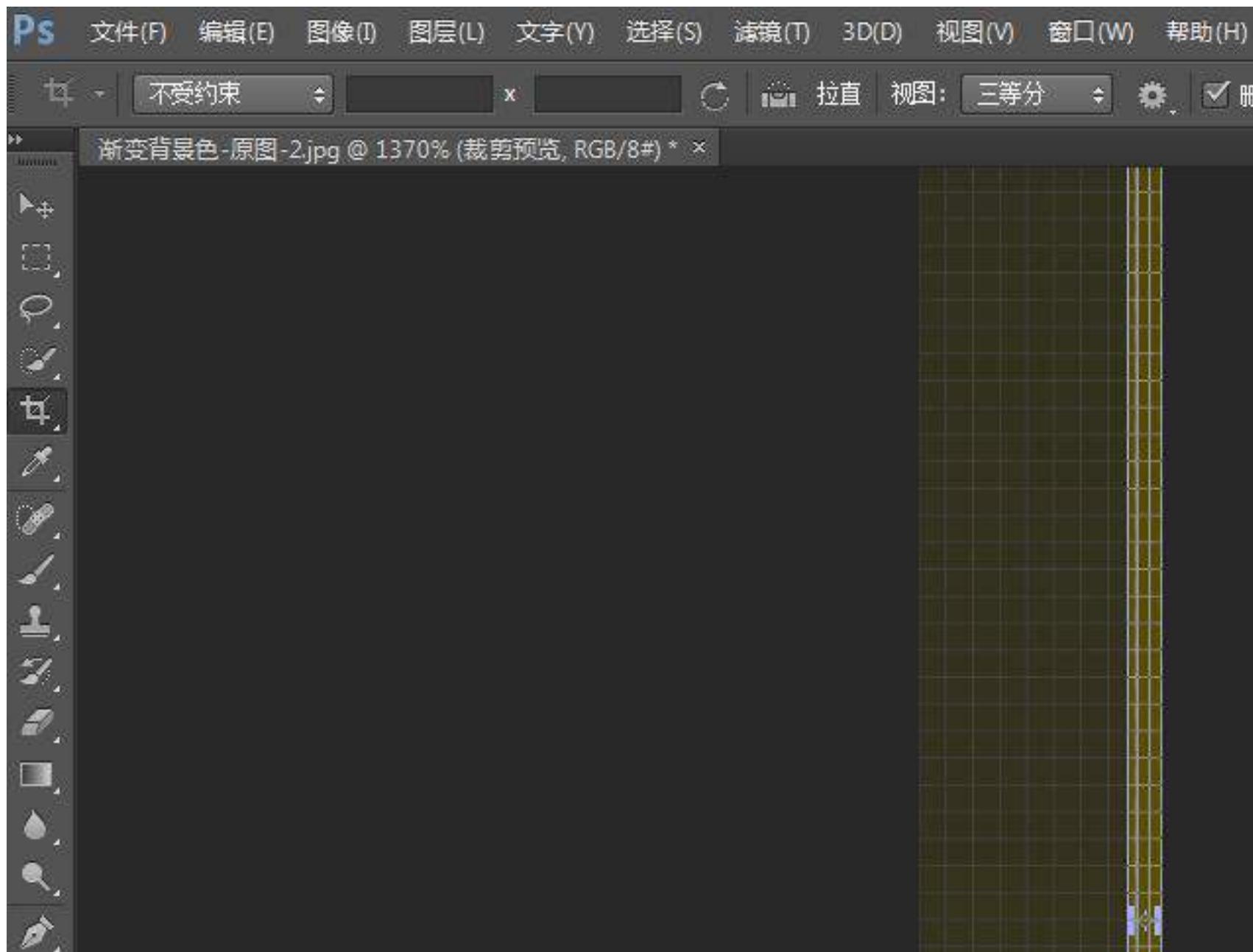
- 新建一个透明的图片
- 文件->置入
- 通过鼠标移动图片至合适位置

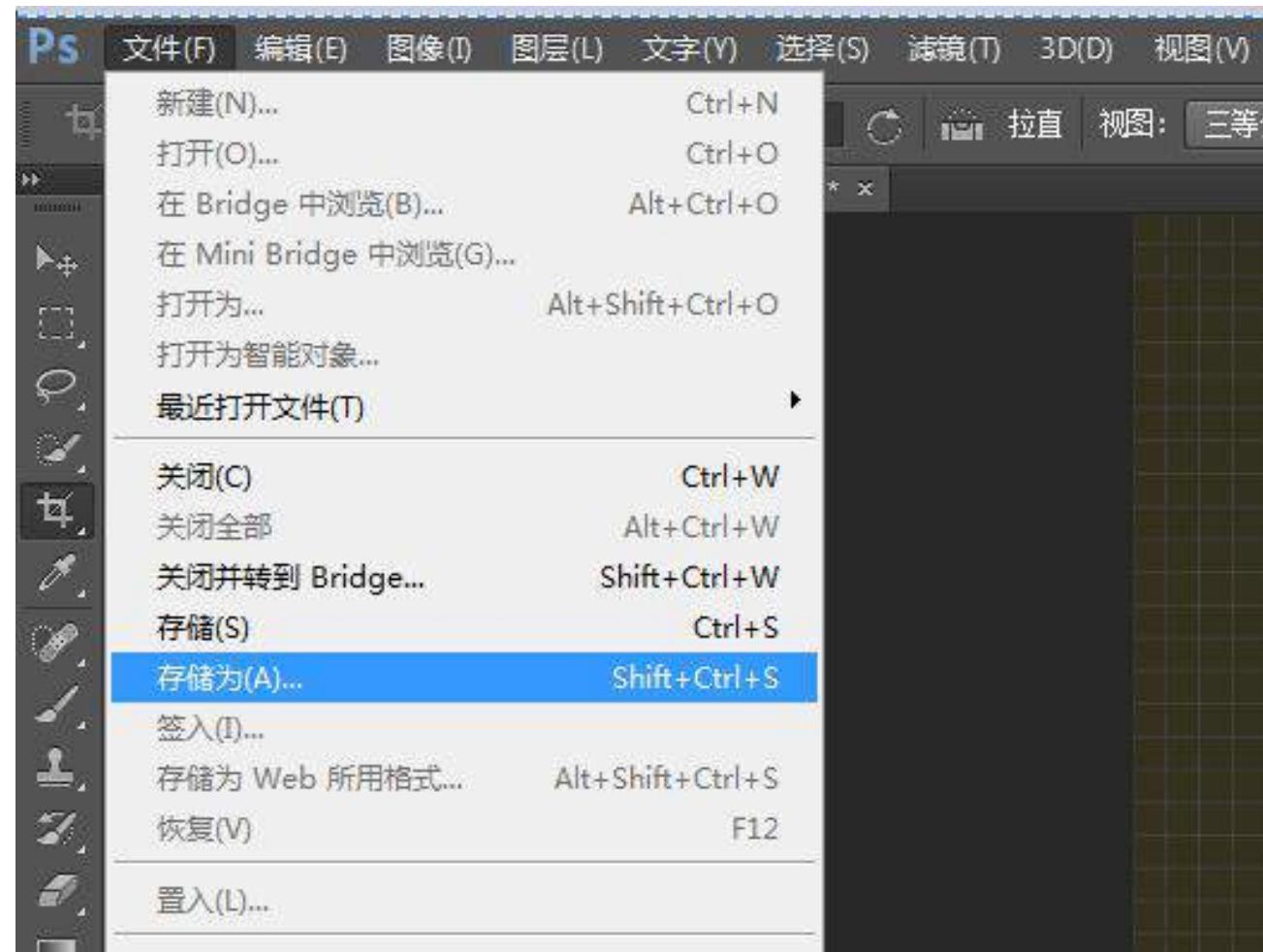


3、裁剪1px背景

- 1、Alt + 鼠标左键放大背景图
- 2、裁剪一像素







Adobe Photoshop CS6 Extended



要在存储之前裁剪图像吗？

裁剪(C)

取消

不裁剪(D)





CSS2 行高 line-height

讲师：许井龙

ngsteel@qq.com



- 在**块级元素**中，行高指定了该块级元素所有的行盒（line box）的最小高度。
- 在**非替换元素**中，行高指定了该元素行盒（line box）的高度，
- 替换元素（img, button）没有行高！！！

扩展知识

1、行盒（line-box）

元素的每一行称为一条line box，它又是由这一行的许多inline-box组成。

2、替换元素

替换元素是浏览器根据其标签的元素与属性来判断显示具体的内容。

比如：<input /> type="text" 的是，这是一个文本输入框，输入新值时，浏览器显示就不一样
(X)HTML中的、<input>、<textarea>、<select>、<object>都是替换元素，这些元素都没有实际的内容。

3、非替换元素

(X)HTML 的大多数元素是不可替换元素，他们将内容直接告诉浏览器，将其显示出来。

比如<p>wanmei.com</p>

浏览器将把这段内容直接显示出来。

- normal
 - 浏览器自己计算，一般是字体大小（font-size）的1.2倍
 - 举例，如果元素的字体大小是16px，此时行高约等于19.2px
- <number>
 - 这是大多数情况下，推荐的做法，可以避免继承造成问题。
 - 举例，`line-height: 2`，如果此时元素的字体大小是20px，那么行高是40px。
- <length>
 - 指定行盒的高度
 - 举例，`line-height: 30px`，此时行高是30px

如果line-height: 未指定具体长度 (px) , 字体大小的改变会影响行高, 此时

1. 如果块元素 (可以设置宽高的元素) 未指定高度, 其高度等于行内元素 (该行内元素是块元素的子元素) 的行高, 如果是多行就是多行行高的累加。
2. 行内元素 (inline) 的行高虽然无法 “撑起” 该行内元素的高度, 但却影响当前行的行高, 进而影响父元素的高度。

- 行高等于元素的高度，可以让文字在块元素的垂直方向对齐（纵向的中间位置）。
- 原因： $(\text{行高} - \text{内容盒子高度}) / 2$ 后分别显示在内容盒子的上下，

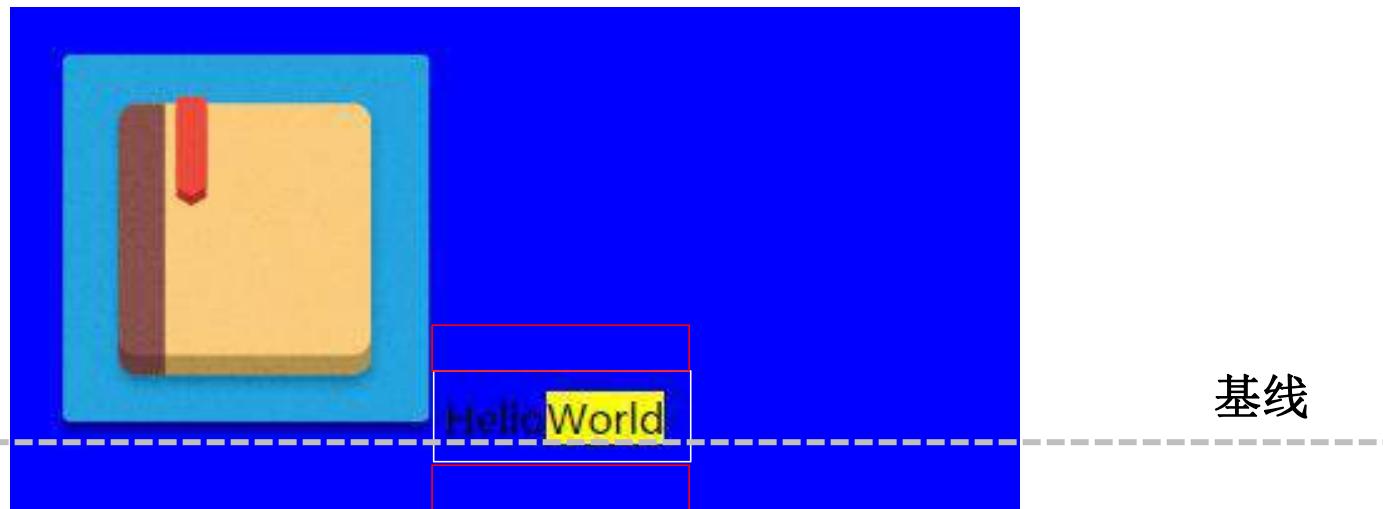
内容盒子



行高 = font-size * 1.2 默认

半行高 = 行高 - 内容盒子的高度/2

- 一行内如果出现替换元素（例如图片），且图片高度过大，文字内容无法在行内中间居中了。



关于基线知识：请参考 CSS 垂直对齐
vertical-align





CSS2 字体

讲师：许井龙

ngsteel@qq.com



- font-family 设置字体系列。
- font-size 设置字体的尺寸。
- font-style 设置字体风格。
- font-variant 以小型大写字体或者正常字体显示文本。
- font-weight 设置字体的粗细。
- font 简写属性。

- font-family 规定元素的字体系列（字体族）。
- font-family 可以把多个字体名称作为一个“回退”系统来保存。如果浏览器不支持第一个字体，则会尝试下一个。
- 例如
 - Microsoft YaHei, tahoma, arial, Hiragino Sans GB, sans-serif

上述字体由操作系统提供。

以**Windows**操作系统为例，字体存储在 **C:\Windows\Fonts** 目录下

Aharoni 粗体	粗体	隐藏	希伯来语	显示	Kivun Computers Ltd
Andalus 常规	常规	隐藏	阿拉伯语	显示	Glyph Systems
Angsana New	常规; 斜体; 粗体; 斜体	隐藏	泰语	文本	Microsoft Corporation
AngsanaUPC	常规; 斜体; 粗体; 斜体	隐藏	泰语	文本	Microsoft Corporation
Aparajita	常规; 粗体; 粗体; 斜体; 斜体	隐藏	梵文	显示	Microsoft Corporation

- 有两种主要类型的字体

1. 衬线字体 (serif) : 在字的笔划开始及结束的地方有额外的装饰，而且笔划的粗细会因直横的不同而有不同。
2. 非衬线字体 (**Sans Serif**) : 没有额外的装饰，笔划粗细大致差不多。



- 设置字体大小
- 默认值：medium。**一般浏览器默认是16px**
- smaller 把 font-size 设置为比父元素更小的尺寸。
- larger 把 font-size 设置为比父元素更大的尺寸。
- *length* 把 font-size 设置为一个固定的值。
- % 把 font-size 设置为基于父元素的一个百分比值。
- inherit 规定应该从父元素继承字体尺寸。

注意：font-size 会影响 line-height 属性 进而影响一行中基线的位置。



注意这里的缝隙

一行中的 行内块元素之间 会有缝隙，如果该元素没有文本内容。我可以设置元素的font-size: 0 解决。但是这回导致该元素所有的文字消失。

导致的原因：

在编码时，``元素换行了。例如

```
<img src= "img/1.png" >  
<img src= "img/2.png" >
```

最佳解决办法

```
<img src= "img/1.png" ><!--  
→<img src= "img/2.png" >
```

- 字体样式
- normal 默认值。浏览器显示一个标准的字体样式。
- italic 浏览器会显示一个斜体的字体样式。
- oblique 浏览器会显示一个倾斜的文字效果。
- inherit 规定应该从父元素继承字体样式。

italic和oblique都是向右倾斜的文字。,

区别在于Italic是指斜体字，而Oblique是倾斜的文字，对于没有斜体的字体应该使用Oblique属性值来实现倾斜的文字效果。

- 小型大写字母字体
- normal 默认值。浏览器会显示一个标准的字体。
- small-caps 浏览器会显示小型大写字母的字体。
- inherit 规定应该从父元素继承 font-variant 属性的值。

- 字体的粗细
- normal 默认值。定义标准的字符。
- bold 定义粗体字符。
- bolder 定义更粗的字符。
- lighter 定义更细的字符。
- 100 ~900 定义由粗到细的字符。400 等同于 normal，而 700 等同于 bold。
- inherit 规定应该从父元素继承字体的粗细。

实际开发中，我们推荐使用关键字方式。因为700 和 900的粗细根本看不出区别

- 常见写法 font: 字体样式 小型大写字母 字体粗细 字体大小/行高 字体族(多个)

字体/行高 必须在字体族前
行高可以不写，默认是浏览器的1.2倍（不同浏览器实现可能不同）

font: **italic bold small-caps 22px/20px arial, sans-serif;**

这三个样式的五先后顺序要求，
但是必须在字体样式前

字体族永远位于最后

实际开发中，我们推荐简写方式！！！



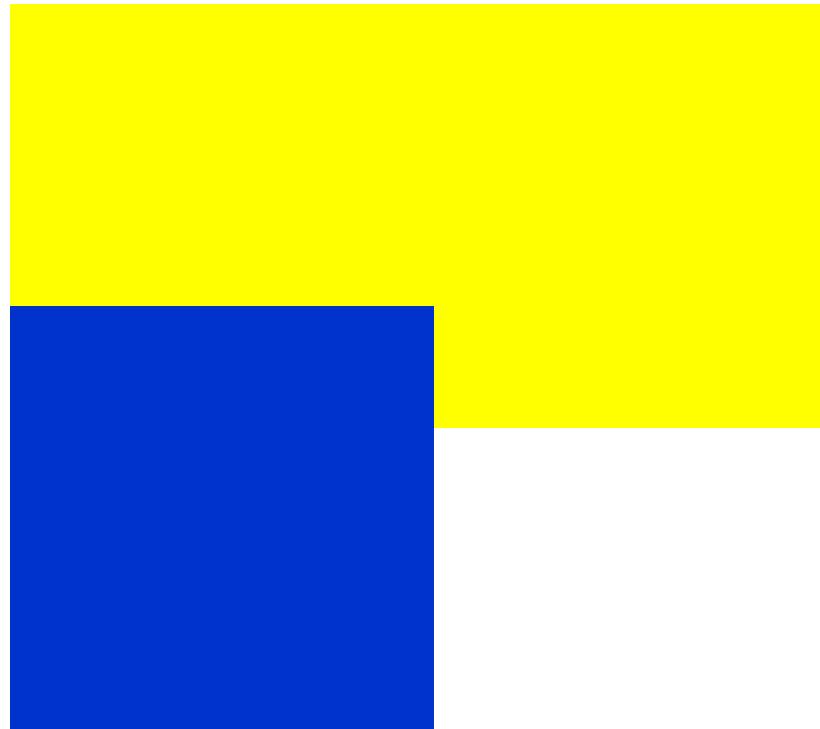


负外边距

讲师：许井龙

ngsteel@qq.com

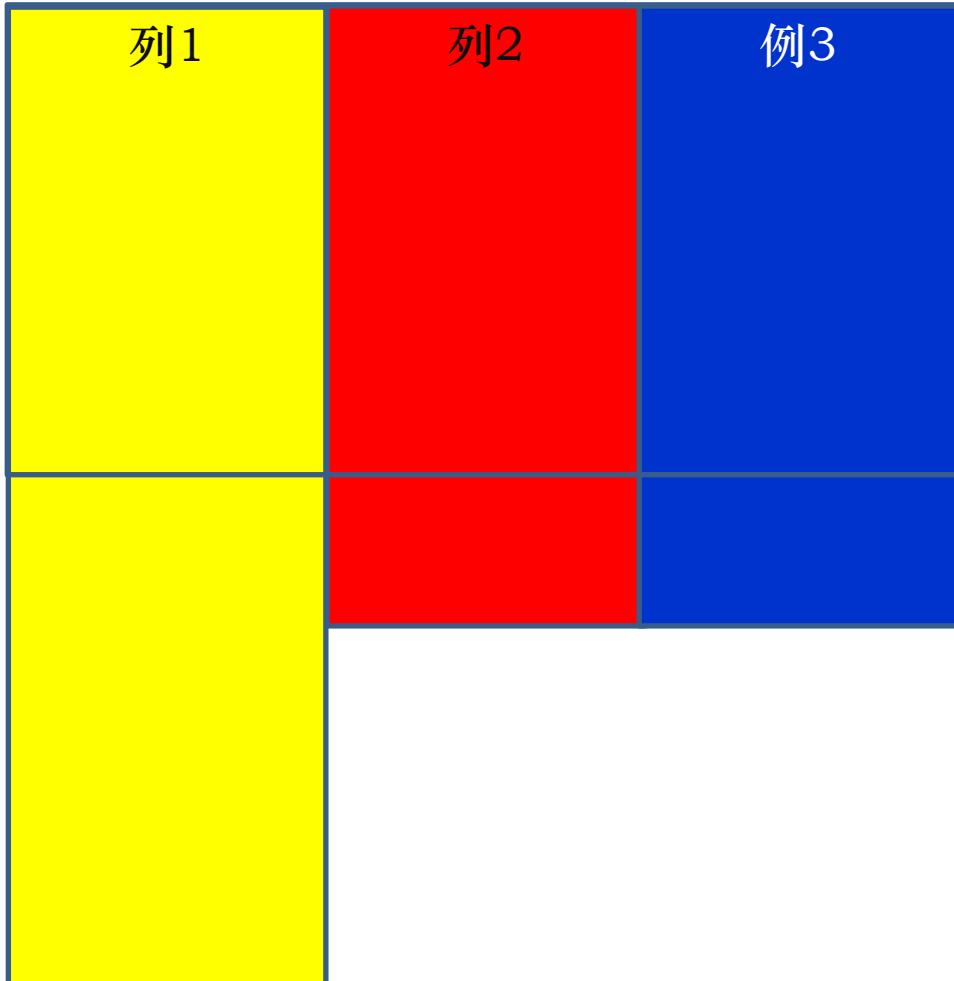
margin-top: 200px



- 浮动造成的层叠
 - 只能浮动在其兄弟元素上
 - 兄弟元素有文本会环绕浮动元素
- 定位（绝对定位，固定定位）的层叠
 - 彻底脱离文档，会对其层级下的元素全覆盖，包裹文本
- 负外边距造成的层叠
 - 本质边界在改变了，效果类似定位造成的层叠。
 - 文本无法被覆盖。

电脑屏幕（视口）

margin-left: 200px



1. 父元素, overflow;
2. 子元素, 底内边距 12000
3. 子元素, 底外边距 -12000



1. `html, body{ height: 100%}`
2. 父元素，高度 100%；
3. 父元素，底**外边距** = 负 固定底部高度
4. 内容区，底**内边距** = 固定底部高度





CSS2 文本

讲师：许井龙

ngsteel@qq.com



1. color 设置文本颜色
2. direction 设置文本方向
3. text-align 对齐元素中的文本
4. text-decoration 向文本添加修饰
5. text-indent 缩进元素中文本的首行
6. text-transform 控制元素中的字母
7. letter-spacing 设置字符间距
8. word-spacing 设置字间距
9. white-space 设置元素中空白的处理方式

- 字体颜色
- color_name 规定颜色值为颜色名称的颜色（比如 red）。
- hex_number 规定颜色值为十六进制值的颜色（比如 #ff0000）。
- rgb_number 规定颜色值为 rgb 代码的颜色（比如 rgb(255,0,0)）。
- 透明色 **rgba(255, 0, 0, .8)**

- 设置文本方向
 - ltr 默认。一行中文本，行内元素，行内块元素 排列方向从左到右。
 - rtl 一行中文本，行内元素，行内块元素 排列方向从右到左。
 - inherit 规定应该从父元素继承 direction 属性的值。



`direction: ltr;`



`direction: rtl;`

注意：

1. 设置**direction**，改变行内元素（inline），文本的对齐，不会改变书写顺序。
2. 设置**direction**，改变行内块元素（inline-block）的排列位置。





طيران الإمارات تطلق خدمة يومية إلى نيوارك عبر أبيها

ستوفر الرحلة الجديدة خدمة على مدار العام من دبي إلى مطار نيوارك ليبرتي الدولي في الولايات المتحدة الأميركية عبر العاصمة اليونانية أثينا اعتباراً من 12 مارس 2017.

[تعرف على المزيد <](#)



阿联酋航空公司推出通过雅典日常服务到纽瓦克

它将通过雅典的希腊首都提供贯穿从迪拜全年纽瓦克自由国际机场在美国
。一个新的飞行服务为2017年3月12日的

[<了解更多](#)

- 快速实现两个按钮的对调，而无需JS。



```
direction: ltr;
```



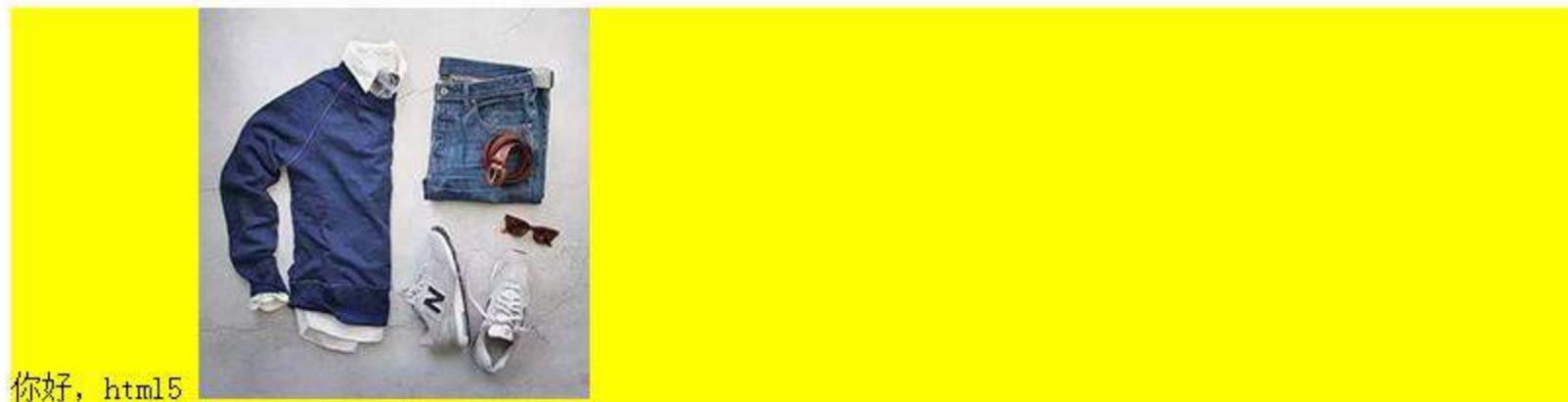
```
direction: rtl;
```

- 规定一行中**文本内容，行内元素，行内块元素**的对齐方式
- 可选值
 - left 把文本排列到左边。默认值：由浏览器决定。
 - right 把文本排列到右边。
 - center 把文本排列到中间。
 - justify 实现两端对齐文本效果。
 - inherit 规定应该从父元素继承 text-align 属性的值。

行内元素 (`display: inline`) : span, a 等

行内块元素 (`display: inline-block`) : img input 等

left 把文本排列到左边



right 把文本排列到右边。

你好，html5



center 把文本排列到中间。

你好，html5



justify 实现两端对齐文本效果。

The line-height property defines the amount of space above and below inline elements. That is, elements that are set to display: inline or display: inline-block. This property is most often used to set the leading for lines of text.

至少一行以上才能看到 两端对齐的效果！！

其本质是“拉伸了空白”的距离

- 文本的修饰。
 - `none` 默认。定义标准的文本。
 - `underline` 定义文本下的一条线。
 - `overline` 定义文本上的一条线。
 - `line-through` 定义穿过文本下的一条线。
 - `inherit` 规定应该从父元素继承 `text-decoration` 属性的值。

Hello World

text-decoration: underline;

~~Hello World~~

text-decoration: line-through;

Hello World

text-decoration: overline;

- <a>元素 默认设置了
 - text-decoration: underline;
- 我们可以设置
 - text-decoration: none; 取消下划线

- 首行缩进

- *length* 定义固定的缩进。默认值：0。
- % 定义基于父元素宽度的百分比的缩进。
- inherit 规定应该从父元素继承 text-indent 属性的值。

实际开发中我们使用cm，确保一个很是的比例!!!

- 控制文本的大小写

- none 默认。定义带有小写字母和大写字母的标准的文本。
- capitalize 文本中的每个单词以大写字母开头。
- uppercase 定义仅有大写字母。
- lowercase 定义无大写字母，仅有小写字母。
- inherit 规定应该从父元素继承 text-transform 属性的值。

Hello World

`text-transform: capitalize;`

hello world

`text-transform: lowercase;`

HELLO WORLD

`text-transform: uppercase;`

- 该属性定义元素中字之间插入多少空白符。针对这个属性，“字” 定义为由空白符包围的一个字符串。如果指定为长度值，会调整字之间的通常间隔；所以，normal 就等同于设置为 0。允许指定负长度值，这会让字之间挤得更紧。
 - normal 默认。规定字符间没有额外的空间。
 - length* 定义字符间的固定空间（允许使用负值）。
 - inherit 规定应该从父元素继承 letter-spacing 属性的值。

- 该属性定义了在文本字符框之间插入多少空间。由于字符字形通常比其字符框要窄，指定长度值时，会调整字母之间通常的间隔。因此，normal 就相当于值为 0。
 - normal 默认。规定字符间没有额外的空间。
 - length* 定义字符间的固定空间（允许使用负值）。
 - inherit 规定应该从父元素继承 letter-spacing 属性的值。





CSS3 背景

讲师：许井龙

ngsteel@qq.com

2016年第一版

- **一、背景的基本属性**
 - **background-position**
 - **background-repeat**
 - **background-attachment**
 - **background-color**
 - **background-image**
- **二、CSS3新增背景属性**
 - **background-origin**
 - **background-clip**
 - **-webkit-background-clip: text** (仅webkit内核)
 - **background-size**

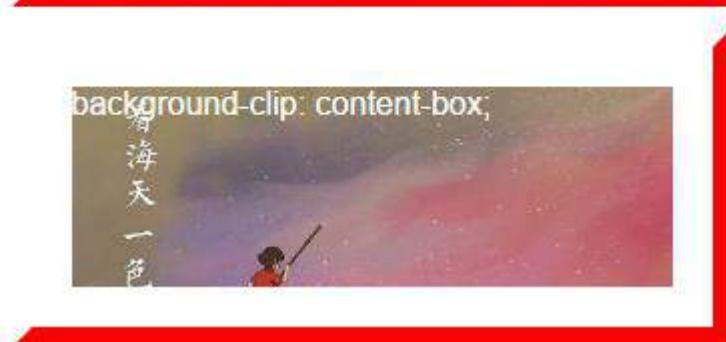
一、背景的基本属性

- background-position: 背景图片位置
 - top right ;
 - 50% 50%;
 - 0 0
- background-repeat:
 - no-repeat;
 - repeat; (默认)
- background-attachment:
 - fixed;
 - scroll
- background-color:
 - #00CCFF;
 - rgb(255, 0, 0);
- background-image:
 - url("logo.png")

- **background-origin: border-box;**
 - 背景图的background-position起始点（默认左上角（0,0））是从元素边框外边缘开始。
- **background-origin: padding-box; (默认值)**
 - 背景图的background-position起始点（默认左上角（0,0））是从元素边框内边缘（padding的外边缘）开始。
- **background-origin: content-box;**
 - 背景图的background-position起始点（默认左上角（0,0））是从元素内容的边框外边缘（padding的内边缘）开始。



- `background-clip: border-box;` (默认值)
 - 元素外边框外的背景图片都将被裁剪掉。
- `background-clip: padding-box;`
 - 元素padding区域外的背景图片都将被裁剪掉。
- `background-clip: content-box;`
 - 元素内容区域外背景图片都将被裁剪掉。



```
-webkit-background-clip: text;  
-webkit-text-fill-color: transparent;
```

Hello World

注意：只有 webkit /blink 内核浏览器实现的使用图片填充文本的效果
该样式在移动端使用较多

- **background-size: auto;** 默认值
 - 保持背景图原始的高度与宽度。就是背景图保持原样。
- **background-size: <length>;** 取整数值，例如px。
 - 设置后，会改变背景大小。如果只设置与一个值，代表背景的宽度，第二个值为auto（根据图片宽度等比计算高度）。
- **background-size: <percentage>;** 使用%0~100%的值。
 - 设置后，其值是相对于背景图所在**元素的宽度**计算。如果元素的宽度是1000px，当background-size: 50% 40%，此时背景图的宽度应该是500 (1000*50%) px, 高度是400 (1000*40%) px。
- **background-size:cover;**
 - 背景图片填满整个元素，此时背景图是不居中显示，如果需要居中显示需要设置background-position: center;。
- **background-size: contain;**
 - 保持背景图片自身的宽高比例。



background-size: 50% 50%;

background-origin: padding-box;

background-size: contain;

- 基于background-size

- 多背景不能简写，需要分别声明，
 - background-image: url1 ~ urln
 - background-repeat: repeat1 ~ repeatn
 - background-position: position1 ~ positionn
 - background-size: size1~sizen
 - background-attachment: attachment1 ~ attachmentn
 - background-clip: clip1 ~ clip2n
 - background-origin: origin1 ~ originn
 - background-color: color1 ~ colorn

- 使用多个图片及线性渐变





CSS3 border 边框

讲师：许井龙

微信: ngsteel



- 一. CSS3 新增的边框样式
- 二. CSS3 边框颜色
- 三. CSS3 图片边框
- 四. CSS3 圆角边框
- 五. CSS3 盒子阴影
- 六. CSS3 元素外轮廓
- 七. CSS3 调整元素盒子大小

- CSS1 与 CSS2 的边框属性

- border-width, 设置元素四个边框的宽度
- border-color, 设置元素四个边框的颜色
- border-style , 元素四个边框的样式。详细参考：[mozilla开发者中心](#)
- 上述三个属性都遵循 “**TRBL原则**” (上, 右, 下, 左) 。
- 简写方式 border: *border-width, border-color, border-style*

- CSS3 新增的边框样式

- `border-(top | right | bottom | left)-color`: 分别设置元素四个边框的颜色。由于浏览器兼容性问题，该属性极少在商业网站中使用。
- `border-image`: 使用图片作为元素的边框。浏览器支持较好，但商业网站应用很少，主要用在个人博客中。
- `border-radius`: 让元素显示圆角效果。浏览器支持较好，应用非常广泛。
- `box-shadow`: 让元素显示阴影效果。应用非常广泛，主要用来呈现元素立体效果。

二、CSS3 边框颜色属性

- CSS3边框 语法：

- border-top-color: [<color>], 上边框颜色
- border-right-color: [<color>], 右边框颜色
- border-bottom-color: [<color>], 下边框颜色
- border-left-color: [<color>], 左边框颜色
- 上述四个样式属性，仍然遵循TRBL原则
- 简写方式，border-color: [<color> | transparent] {1, 4} | inherit,

特别说明：

- 1、该样式可以应用到所有的元素，甚至伪元素。例如::first-letter。
- 2、该样式具有动画效果。

二、CSS3边框颜色

- 为边框设置多个颜色 (仅Firefox浏览器支持)
 - -moz-border-top-colors: [color]{1,n}
 - -moz-border-right-colors: [color]{1,n}
 - -moz-border-bottom-colors: [color]{1,n}
 - -moz-border-left-colors: [color]{1,n}
 - 每个属性后面可以设置N个合法颜色值，每个颜色占1px。如果元素的边框设置了20px，而颜色只设置了10个，剩下的10px都使用最后一个颜色值。
- 注意：
- 上述属性仅在Firefox3.0以上有效，它不是标准语法。而且目前仍在实验阶段，实际使用一定要慎重。
 - 上述样式的优先级比 border-[top | right | bottom | left]-color 优先级高。
 - 不支持动画效果

示例代码：立体效果的边框

```
-moz-border-bottom-colors: #000 #111 #222 #333 #444 #555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee #fff;  
-moz-border-top-colors: #000 #111 #222 #333 #444 #555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee #fff;  
-moz-border-left-colors: #000 #111 #222 #333 #444 #555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee #fff;  
-moz-border-right-colors: #000 #111 #222 #333 #444 #555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee #fff;
```

需要配合
border: 10px solid;
一起使用。

注意：该样式仅在Firefox浏览器下可使用

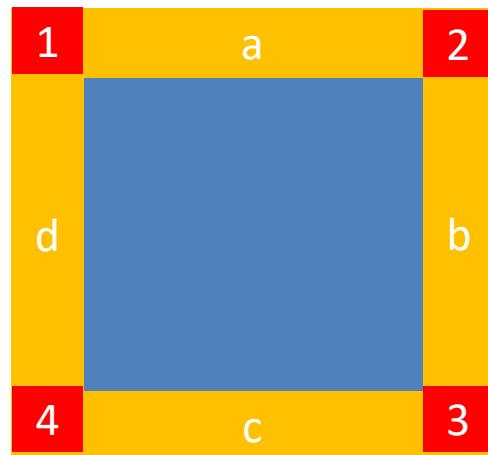
图片边框 `boder-image`

- border-image-source: url(图片路径)
 - 设置图片路径。

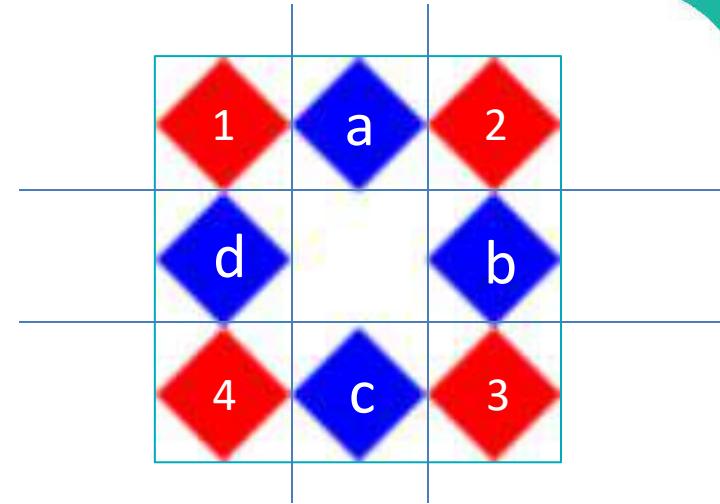
图片边框样式生效的前提：必须设置 border 的宽度 和 样式

例如，border: 1px solid 或者 border: 2px dotted 等等

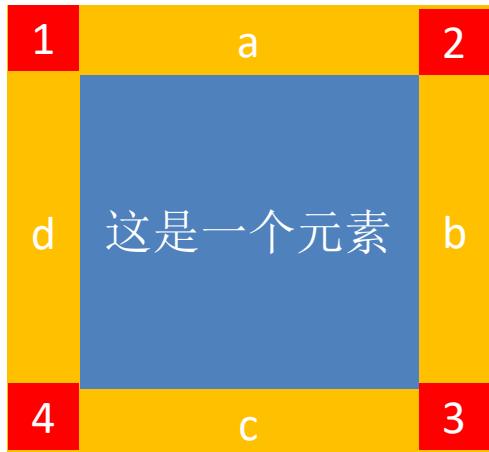
- border-image-slice: [<percentage> | <number>] {1,4} && fill。
- 图片切割的方式。让元素的边框的 **四个角，四条边** 显示切割后的图片。
- 可选值
 1. 关键字: fill (默认值) **图片填充边框的四个角。**
 2. 数字{1, 4} (不需要写单位, 默认为px)
 3. 百分比值 {1, 4} --- 基于图片的宽度或者高度



- border-image-slice:
- 使用数值
 - 26
 - 等价于 26 26 26 26
- 切割顺序及位置
 1. 距离图片上边 26px 位置切割
 2. 距离图片右边 26px 位置切割
 3. 距离图片下边 26px 位置切割
 4. 距离图片左边 26px 位置切割
- 经过上述切割
 - 我们已经把一张图片切割成了9份
 - 按照其编号分别对应元素边框的4个角、4个边（中间的图不需要）



图片原始大小
78px * 78px



- border-image-width: [<length>] {1,4}
 - 用来表示背景图片显示的大小。
 - 与border-width设置效果相同，**但是不会导致元素的实际大小改变。**
- 样式值
 - 10px 四个图片边框都是10px
 - 10px 20px 上下边框 10px , 左右边框 20px
 - 10px 20px 30px 上边框 10px ,左右边框 20px, 下边框 30px
 - 10px 20px 30px 40px 上边框 10px ,右边框 20px, 下边框 30px, 左边框 40px

- border-image-repeat: [stretch | round | repeat]
 - stretch: 默认值 四个边的图片被拉伸
 - round: 填满 四个边的图片重复平铺，**确保图片完整性。**
 - repeat: 重复 四个边的图片重复平铺

- /* border-image-width: all */
 - border-image-width: 3;
- /* border-image-width: vertical horizontal */
 - border-image-width: 2em 3em;
- /* border-image-width: top horizontal bottom */
 - border-image-width: 5% 15% 10%;
- /* border-image-width: top right bottom left */
 - border-image-width: 5% 2em 10% auto;

- border-image-outset: [<length>] {1,4}
- 图片外边框的位置，只能为整数，位移不会改变盒子的大小。
- 样式值
 - 10px 四个图片边框都向外移动10px
 - 10px 20px 上下边框向外移动10px，左右边框向外移动20px
 - 10px 20px 30px 上边框向外移动10px ,左右边框向外移动20px, 下边框向外移动30px
 - 10px 20px 30px 40px 上边框向外移动10px ,右边框向外移动20px, 下边框向外移动 30px, 左边框向外移动 40px

- 简写方式: border-image: <border-image-source>||<border-image-slice> [<border-image-width> || <border-image-repeat>]

圆角 `boder-radius`

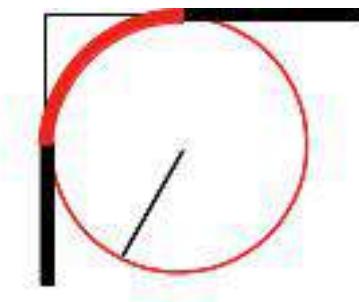
- border-radius: none | <length> {1, 4} [/ <length> {1, 4}] ?

1. 如果有反斜杠，前面设置的元素圆角水平方向半径，后面是垂直方向半径。
2. 如果没有反斜杠，表示圆角水平、垂直方向半径相同。



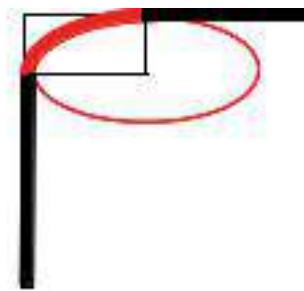
无圆角

border-radius: none



圆形圆角

border-radius: 50px



椭圆形圆角

border-radius: 50px 15px

- border-radius:
 - <length>{1} , 四个角的值相同，也就是4个圆角。
- border-radius:
 - <length>{2} , top-left等于bottom-right，并且取第一个值。top-right等于bottom-left，取第二个值。
- border-radius:
 - <length>{3} , top-left取第一个值，top-right和bottom-left取第二个值，bottom-right取第三个值。
- border-radius:
 - <length>{4} , 四个角取不同的值，top-left取第一个，top-right取第二个，bottom-right取第三个，bottom-left取第四个。

1. 如果不想给元素设置圆角，只需要 border-radius: 0 即可。
2. border-radius: 50%，如果元素是正方形，显示圆形
3. border-radius: 50%，如果元素是长方形，显示椭圆

- 属性box-shadow property设置盒子的阴影
- box-shadow:
 - 参数1：投影方式。默认外阴影， inset是内阴影。
 - 参数2： X轴偏移量， 可以是负值
 - 参数3： Y轴偏移量， 可以是负值
 - 参数4： 阴影模糊半径
 - 参数5： 阴影扩展半径， 可以是负值
 - 参数6： 阴影颜色；

- none: 默认值，元素不设置阴影。
- inset: 阴影类型，可选。如果不设置投影方式外部阴影，如果设置inset就是内 阴影
- x-offset: 阴影水平偏移。正值，阴影在元素右侧，负值在元素左侧。
- y-offset: 阴影垂直偏移。正式，阴影在元素底部，负值在元素顶部。
- blur-radius: 阴影的模糊半径，可选。只能为正值，如果设置为0，阴影不具备模糊效果。
- spread-radius: 阴影扩展半径。可选参数。其值可正可负。如果取正值，整个阴影都延展扩大，反之取负值，则整个阴影都缩小。
- color: 阴影的颜色。 可选参数

五、CSS3 盒子阴影属性

- 单边实影：只需要调整 x-offset 或者 y-offset 即可。
 - box-shadow: 0 -2px #666 上阴影
 - box-shadow: 2px 0 #666 右阴影
 - box-shadow: 0 2px #666 下阴影
 - box-shadow: -2px 0 #666 左阴影
- 单边阴影：调整 x-offset 或者 y-offset 设置偏移，设置模糊半径实现模糊效果。
 - box-shadow: 0 -2px 5px #666 上模糊阴影
 - box-shadow: 2px 0 5px #666 右模糊阴影
 - box-shadow: 0 2px 5px #666 下模糊阴影
 - box-shadow: -2px 0 5px #666 左模糊阴影

- 单边阴影：设置阴影深度。
 - box-shadow: 0 -2px 5px 6px #666 上模糊阴影
 - box-shadow: 2px 0 5px 6px #666 右模糊阴影
 - box-shadow: 0 2px 5px 6px #666 下模糊阴影
 - box-shadow: -2px 0 5px 6px #666 左模糊阴影
 - 阴影扩展半径是正值，与设置相同大小边框（border）的效果相同

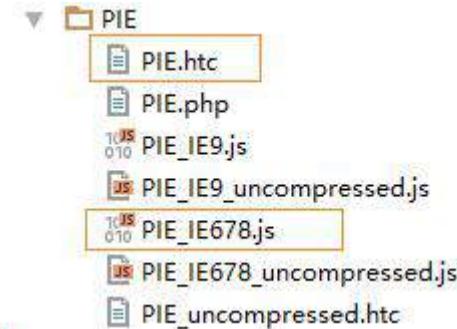
- **outline:** 元素轮廓
 - 语法 [<'outline-color'> || <'outline-style'> || <'outline-width'>]
 - outline-color 外轮廓颜色
 - outline-style 外轮廓样式
 - outline-width 外轮廓宽度
- **outline-offset:** 外轮廓偏移量，不会导致盒子大小改变。
 - 正值：外偏移 outline-offset: 10px
 - 负值：内偏移 outline-offset: -10px
- 知识点：
 - 外轮廓不会改变元素盒子大小

- **resize:**
 - none: 用户不能调整元素宽高。
 - both: 用户可以调整元素宽高。
 - horizontal: 用户只能调整元素宽度
 - vertical: 用户只能调整元素高度
 - inherit: 默认继承
 - 注意事项: **resize 必须搭配 overflow: auto;**

解决IE低版本兼容性问题

- border-radius: 圆角
- box-shadow: 阴影
- border-image: 图片边框

- 第一步：下载PIE.js项目压缩文档
- 第二步：解压缩文件。把PIE



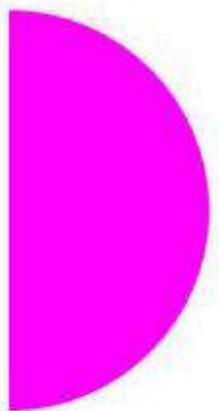
```
<!--[if lte IE 8]>
<script type="text/javascript" src="../PIE/PIE_IE678.js"></script>
<![endif]-->

<!--[if IE 9]>
<script type="text/javascript" src="../PIE/PIE_IE9.js"></script>
<![endif]-->
```

border-radius: 0 100px 100px 0;

behavior: url(../PIE/PIE.htc);

- 第三步：编写CSS3样式
- 第四步：behavior: url(path/to/pie_files/PIE.htc);
- 第五步：在IE下查看



模式

文档模式

通过 F12 开发者工具栏

浏览器配置文件

用户代理字符串





HTML5 classList

讲师：许井龙

ngsteel@qq.com



- 在HTML5 API里，页面DOM里的每个节点上都有一个classList对象，程序员可以使用里面的方法新增、删除、修改节点上的CSS类（即class属性值）。
- 使用classList，程序员还可以用它来判断某个节点是否被赋予了某个CSS类。

- HTML5 DOM的 classList 对象的 length , 返回class属性值的个数

```
<div id="div01" class="box box-3d box-green"></div>
```

```
<script type="text/javascript">  
    var div01 = document.querySelector("#div01");  
    alert(div01.classList.length); //返回3  
</script>
```

HTML

```
<div id="div01" class="box box-3d box-green"></div>
```

JS

```
var div01 = document.querySelector("#div01");
div01.classList.add("box-blue");
```

JS执行后

```
<div id="div01" class="box box-3d box-green box-blue"></div>
```

HTML

```
<div id="div01" class="box box-3d box-green"></div>
```

JS

```
var div01 = document.querySelector("#div01");
div01.classList.remove("box-green");
```

JS执行后

```
<div id="div01" class="box box-3d"></div>
```

HTML

```
<div id="div01" class="box box-3d box-green"></div>
```

JS

```
var div01 = document.querySelector("#div01");
console.log(div01.classList.contains("box-green"));
```

控制台输出结果： true

HTML

0 1 2

```
<div id="div01" class="box box-3d box-green"></div>
```

JS

```
var div01 = document.querySelector("#div01");
console.log(div01.classList.item(2));
```

控制台返回 box-green

如果class属性存在指定了类名删除该类名，否则添加

HTML

```
<div id="div01" class="box box-3d box-green"></div>  
<button id="toggleBtn">切换</button>
```

JS

```
var div01 = document.querySelector("#div01");  
var toggleBtn = document.querySelector("#toggleBtn");  
toggleBtn.onclick = function(){  
    div01.classList.toggle("box-green")  
}
```

第奇数次单击

```
<div id="div01" class="box box-3d"></div>
```

第偶数次单击

```
<div id="div01" class="box box-3d box-green "></div>
```



HTML5 dataset 自定义属性

讲师：许井龙

ngsteel@qq.com

- 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。

```
<div id="div01" data-goods-desc="我是自定义属性的值"></div>
```

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

```
<div id="div01" data-goods-desc="我是自定义属性的值"></div>
```

```
<script type="text/javascript">  
    //获取DOM  
    var div01 = document.querySelector("#div01");  
    //DOM里的dataset对象，包含了所有的自定义属性  
    console.log(div01.dataset.goodsDesc);  
</script>
```

HTML

data-goods-desc

JS

dataset.goodsDesc

```
<div id="div01" data-goods-desc="我是自定义属性的值"></div>
```

```
<script type="text/javascript">  
    //获取DOM  
    var div01 = document.querySelector("#div01");  
    //DOM里的dataset对象，包含了所有的自定义属性  
    div01.dataset.goodsDesc = "新属性值";  
</script>
```

JS代码执行后

```
<div id="div01" data-goods-desc="新属性值"></div>
```

```
<div id="div02"></div>
```

```
var div01 = document.querySelector("#div02");
div01.dataset.myBirthday = "1981";
```

JS代码执行后

```
<div id="div02" data-my-birthday = "1981"></div>
```

JS

HTML

dataset.myBirthday → data-my-birthday



扩展用法：JSON格式字符串

实际开发存储JSON格式字符串

```
<div id="div02" data-goods-desc='{"name": "tom", "age": 18}'></div>
```

JS代码，获取JSON格式字符串

```
var div01 = document.querySelector("#div02");
var dataVal = div01.dataset.goodsDesc;
dataVal = JSON.parse(dataVal);
console.log(dataVal.name);
```

JSON格式字符串一定要使用双引号

返回的是字符串，一定要转成JSON才能使用

```
<div id="div02" data-goods-desc=""></div>
```

JS代码，添加JSON格式字符串

```
var jsonObj = {"name": "tom", "age": 18};  
var div01 = document.querySelector("#div02");  
var str = JSON.stringify(jsonObj);  
div01.dataset.goodsDesc = str;
```

注意：一定要把JSON对象转换成JSON格式字符串！！！否则不能添加

代码执行后，

```
<div id="div02" data-goods-desc='{"name": "tom", "age": 18}'></div>
```



CSS3 Animation 动画

讲师：许井龙

ngsteel@qq.com

2016年第一版

属性	描述
@keyframes	规定动画
animation-name	规定 @keyframes 动画的名称。
animation-duration	规定动画完成一个周期所花费的秒或毫秒。默认是 0。
animation-timing-function	规定动画的速度曲线。默认是 "ease"。
animation-delay	规定动画何时开始。默认是 0。
animation-iteration-count	规定动画被播放的次数。默认是 1。
animation-direction	规定动画是否在下一周期逆向地播放。默认是 "normal"。
animation	所有动画属性的简写属性， 除了 animation-play-state 属性。

定义动画的帧

```
from{} = 0%{}
```

```
to{}=100%{}
```

```
@keyframes anim1 {  
    from{  
        background-color: red;  
    }  
    to{  
        background-color: blue;  
    }  
}
```

```
@keyframes anim2 {  
    0%{  
        background-color: red;  
    }  
    10%{  
        background-color: green;  
    }  
    100%{  
        background-color: blue;  
    }  
}
```

```
@keyframes anim3 {  
    from{  
        background-color: red;  
    }  
    10%{  
        background-color: green;  
    }  
    to{  
        background-color: blue;  
    }  
}
```

```
.box{  
    width: 100px;  
    height: 100px;  
    animation-name: anim1; ←  
    animation-duration: 2s;  
}  
  
@keyframes anim1 {  
    from{  
        background-color: red;  
    }  
    10%{  
        background-color: green;  
    }  
    to{  
        background-color: blue;  
    }  
}
```

值	描述
n	定义动画播放次数的数值。
infinite	规定动画应该无限次播放。

值	描述
normal	默认值。动画应该正常播放。
alternate	动画应该轮流反向播放。

animation-play-state 属性

值	描述
paused	规定动画已暂停。
running	规定动画正在播放。

值	描述
none	不改变默认行为。
forwards	当动画完成后，保持最后一个属性值（在最后一个关键帧中定义）。
backwards	在 animation-delay 所指定的一段时间内，在动画显示之前，应用开始属性值（在第一个关键帧中定义）。
both	向前和向后填充模式都被应用。

实例：淡入，淡出效果



属性	可能的值
column-width	column-width: 200px
column-count	column-count: 5
columns	columns: 300px 6
column-gap	column-gap: 20px
column-fill	column-fill: balance
column-rule	
column-rule-color	
column-rule-style	
column-rule-width	

column-span	column-span: all
-------------	------------------

描述
规定列的宽度。配合column-count一起使用。 注意：实际的列宽可能比指定的列宽大（用来填充剩余空间），或者小（剩余空间不够时，列会被压缩）
规定显示的列数。
简写属性，同时设置column-width 和 column-count
规定列之间的间隔距离。
生效条件：设置父元素的高度
规定“元素的内容”在每列中如何填充。 如果值是balance（默认值），表示均匀分布到每列。 如果值是auto，逐列填充。
设置所有column-rule-* 属性的简写属性。（与border的设置完全相同）
规定列之间规则的颜色。
规定列之间规则的样式。
规定列之间规则的宽度。

子元素中设置
规定元素应该横跨的列数。
跨所有列使用 all



过渡 transition

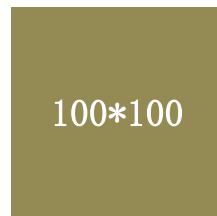
讲师：许井龙

微信：ngsteel



- 一. 过渡简介
- 二. 过渡属性
 - 1. 具有过渡效果的CSS样式属性
 - 2. 过渡持续的时间
 - 3. 过渡延迟时间
 - 4. 过渡效果
 - 关键字
 - 三次贝塞尔函数及贝塞尔曲线
- 三. 过渡简写及过渡覆盖问题
- 四. 单向过渡VS双向过渡
- 五. 过渡触发
- 六. 隐式过渡
- 七. 事件：过渡结束
- 八. 练习：过渡的综合运用

- 什么是过渡 (transition) ?
 - 过渡(transition)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。



需求说明：

- 当鼠标悬浮在棕色背景的矩形时。
- 仅让其宽度变成原来的2倍。
- 变化的过程持续3秒钟。

1. 设置过渡触发的条件

*:hover*状态伪类

2. 指定过渡效果的CSS样式

width

3. 设置过渡的持续时间

3s

这也是实际开发中，实现过渡效果的关键步骤！

- 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
 1. **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）**
 - 1. transition-property: width
 2. **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
- 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的值即不是数值，也无法转为数值。**
 - transition-property: visibility

- 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 多个过度时间使用逗号分隔，通常配合控制多个过渡属性。

transition-property: pro1, pro2, pro3, pro4, pro5, pro6, pro7

transition-duration: 2s

所有属性的过渡时间都是2s

transition-property: pro1, pro2, pro3, pro4, pro5, pro6, pro7

transition-duration: 2s, .3s

过渡时间为 2s 的属性是 pro1 pro3 pro5 pro7

过渡时间为 .3s 的属性是 pro2 pro4 pro6

transition-property: pro1, pro2, pro3, pro4, pro5, pro6, pro7

transition-duration: .2s, 5s, .1s

过渡时间为 .2s 的属性是 pro1 pro4 pro7

过渡时间为 5s 的属性是 pro2 pro5

过渡时间为 .1s 的属性是 pro3 pro6

- 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

```
transition-property: pro1, pro2, pro3, pro4, pro5, pro6,  
pro7
```

```
transition-delay: 2s
```

所有属性的过渡动画都会延时2秒执行

```
transition-property: pro1, pro2, pro3, pro4, pro5, pro6,  
pro7
```

```
transition-delay: 2s, .3s
```

过渡动画都会延时 2s 的属性是 pro1 pro3 pro5 pro7

过渡动画都会延时 .3s 的属性是 pro2 pro4 pro6

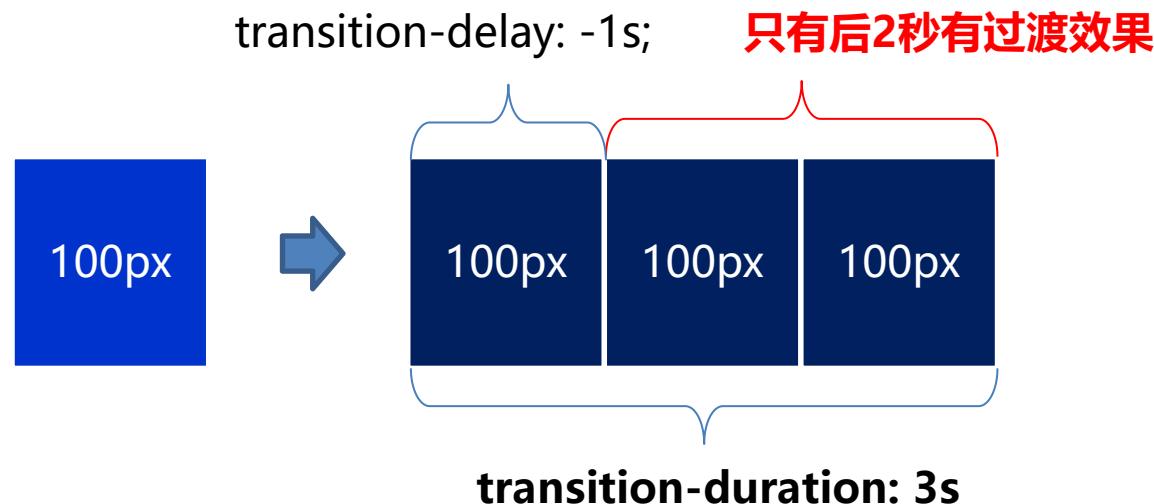
```
transition-property: pro1, pro2, pro3, pro4, pro5, pro6, pro7
```

```
transition-delay: .2s, 5s, .1s
```

过渡动画都会延时 .2s 的属性是 pro1 pro4 pro7

过渡动画都会延时 5s 的属性是 pro2 pro5

过渡动画都会延时 .1s 的属性是 pro3 pro6



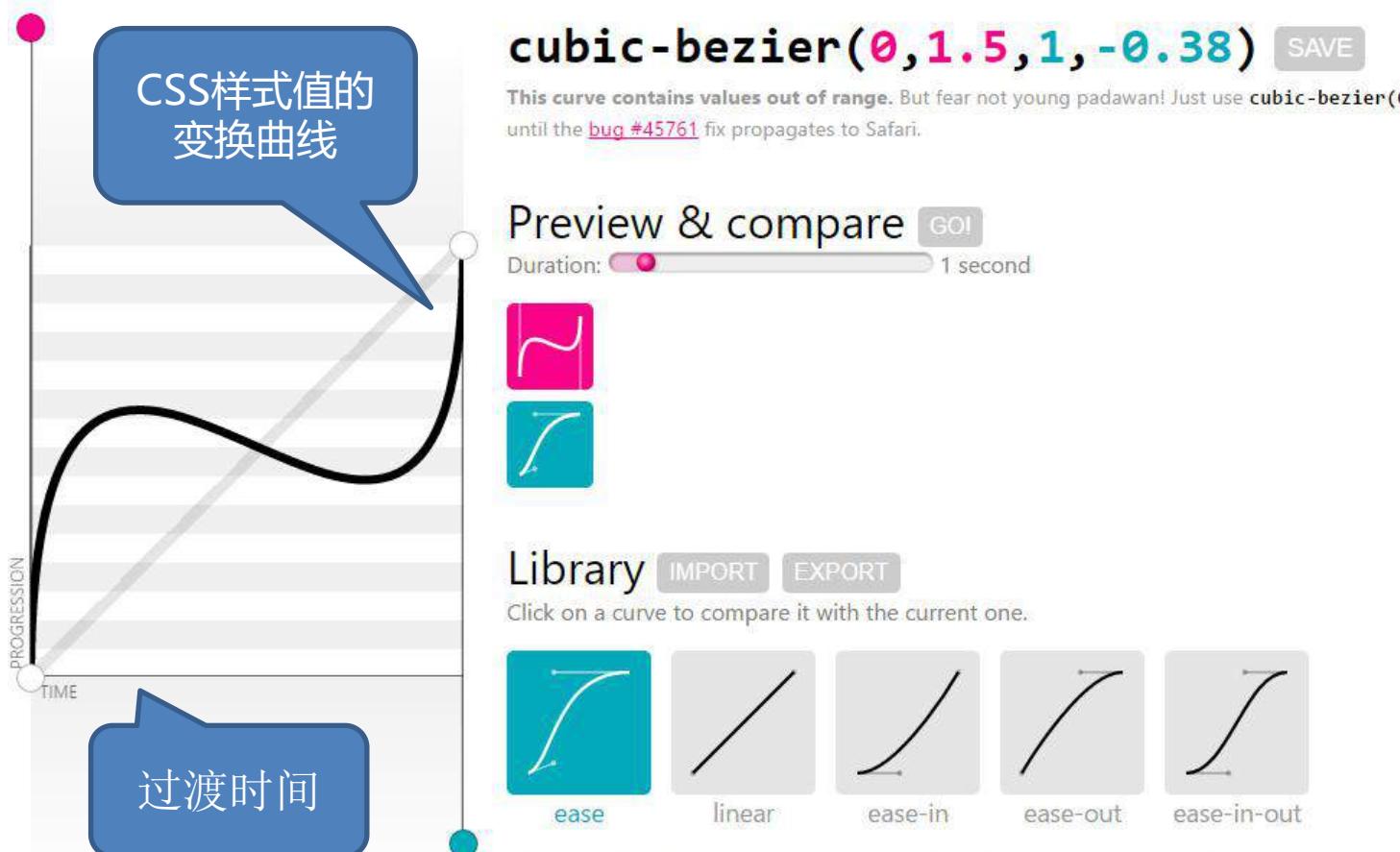
此时我们看到的过渡动画持续时间2秒

- 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关键字，默认
 - transition-timing-function: ease

值	描述
linear	规定以相同速度开始至结束的过渡效果（等于 cubic-bezier(0,0,1,1)）。
ease	规定慢速开始，然后变快，然后慢速结束的过渡效果（cubic-bezier(0.25,0.1,0.25,1)）。
ease-in	规定以慢速开始的过渡效果（等于 cubic-bezier(0.42,0,1,1)）。
ease-out	规定以慢速结束的过渡效果（等于 cubic-bezier(0,0,0.58,1)）。
ease-in-out	规定以慢速开始和结束的过渡效果（等于 cubic-bezier(0.42,0,0.58,1)）。
cubic-bezier(<i>n,n,n,n</i>)	在 cubic-bezier 函数中定义自己的值。可能的值是 0 至 1 之间的数值。

- 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。

cubic-bezier(0,1.5,1,-0.38)



cubic-bezier(1,1.3,0,-0.38)

cubic-bezier(1,1.3,0,-0.38) SAVE

This curve contains values out of range. But fear not young padawan! Just use `cubic-bezier(1,1,0,0)` as well for Webkit until the [bug #45761](#) fix propagates to Safari.

Preview & compare GO!
Duration: 1 second



Library IMPORT EXPORT

Click on a curve to compare it with the current one.



Tip: Right click on any library curve and select "Copy Link Address" to get a permalink to it which you can share with others

cubic-bezier(.66,-0.53,.41,1.52)



cubic-bezier(.66,-0.53,.41,1.52) SAVE

This curve contains values out of range. But fear not young padawan! Just use `cubic-bezier(.66,0,.41,1)` as well for Webkit until the [bug #45761](#) fix propagates to Safari.

Preview & compare GO!

Duration: 1 second

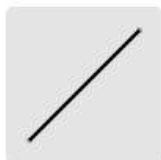


Library IMPORT EXPORT

Click on a curve to compare it with the current one.



ease



linear



ease-in



ease-out



ease-in-out

Tip: Right click on any library curve and select "Copy Link Address" to get a permalink to it which you can share with others

- 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。

```
/*
    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，过渡时间为零，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;

```
/*
transition:width 3s, all 0s;
```

• 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）

单向过渡VS双向过渡

```
.box {  
    width: 100px;  
    height: 100px;  
    background-color: #00f;  
}
```

开始
样 式

```
.box {  
    width: 100px;  
    height: 100px;  
    background-color: #00f;  
transition: 3s;  
}
```

单向过渡 

VS

  双向过渡

```
body:hover .box {  
    width: 400px;  
transition: 3s;  
}
```

结 束
样 式

```
body:hover .box {  
    width: 400px;  
}
```

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

- Firefox 与 Chrome 会触发隐式过渡。

```
.box {  
    width: 5em;  
    height: 5em;  
    background-color: #00f;  
    color: #fff;  
    font-size: 20px;  
}  
  
body:hover .box {  
    font-size: 40px;  
    transition: font-size 3s;  
}
```

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

- 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);
```

过渡结束事件对象 相关属性

事件属性	类型	描述
target	EventTarget	触发过渡结束事件的源
type	DOMString	事件的类型
bubbles	Boolean	是否支持事件冒泡
cancelable	Boolean	是否取消事件冒泡
propertyName	DOMString	与过渡关联的CSS属性名
elapsedTime	Float	过渡的持续时间（如果延迟时间为正数，含延迟时间）

重点总结

1. 设置过渡的持续时间

- a. 默认值为 0s，即默认没有过渡
- b. 只要该值大于 0s 即可。

2. 设置过渡的触发条件

- a. 状态伪类触发
- b. JS触发
- c. @media触发

1. 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）

2. 过渡简写导致样式覆盖问题

- ✓ 逗号后面的样式会覆盖前面的样式。

- 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)





transform 2D

讲师：许井龙

微信 : ngsteel



1. transform 2d 简述
2. 缩放函数 scale()
3. 变型中心点
4. 倾斜函数 skew()
5. 位移函数 translate()
6. 旋转函数 rotate()
7. 变形函数组合使用

- 变形(transform)是**CSS3新增的一个样式**，该样式配合4个变形函数一起使用，可以让元素发生如下变化
 1. 缩放 scale()
 2. 倾斜 skew()
 3. 位移 translate()
 4. 旋转 rotate()
- 当然我们也可以组合使用上述函数
- 配合过渡(transition)可以轻松实现很多“有规律的”酷炫特效。

- 语法介绍
 - `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。
- 如何使用
 - `transform: scale(x, y)`
 - `transform: scaleX(n)`
 - `transform: scaleY(n)`
- 参数说明
 - ① $0 \leq (x, y, n) < 1$ 缩小
 - ② $(x, y, n) = 1$ 默认值
 - ③ $(x, y, n) > 1$ 放大

- 普通的元素放大，设置元素的宽度(width)与高度(height)，会有如下问题，
 1. 默认基于元素左上角缩放。
 2. 会影响兄弟元素的位置。
 3. 如果父元素未设置大小，会影响父元素的大小。
- 缩放函数缩放元素的特点，
 1. 只能对块元素设置缩放 (确切的说所有的变换函数都只能设置在块元素中)
 2. 可以设置在元素内任一点进行缩放。也被称作变形的中心点
 3. 不会影响兄弟元素的位置。
 4. 如果父元素未设置大小，不会影响父元素的大小。

- 使用场景
 - 放大一个元素的同时，需要放大其内容及其子元素
 - 放大一个元素，不影响现有布局。
- 实际运用
 - 例如商品图片放大

- 语法介绍
 - transform-origin: x-axis y-axis z-axis;
- 参数说明
 - x-axis 定义视图被置于 X 轴的何处。可能的值：
 - left
 - center
 - right
 - length
 - %
 - y-axis 定义视图被置于 Y 轴的何处。可能的值：
 - top
 - center
 - bottom
 - length
 - %
 - z-axis 定义视图被置于 Z 轴的何处。可能的值：
 - length
- 扩展知识：
 - z-axis 的值不能使用%和关键字
 - 2D变换不需要考虑 z-axis

说明：
x-axis X 轴
y-axis Y 轴
z-axis Z 轴

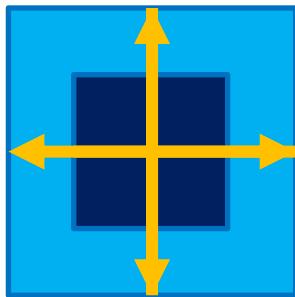
变换中心点

transform-origin: center

transform-origin: 50%

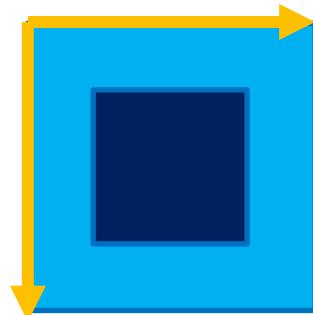
transform-origin: 100px 100px

默认值



transform-origin: top left

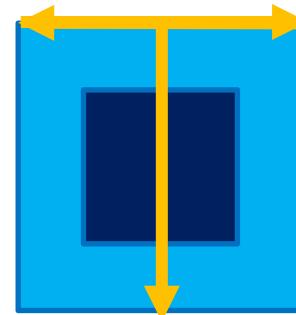
transform-origin: 0 0



transform-origin: 50% 100%

transform-origin: top

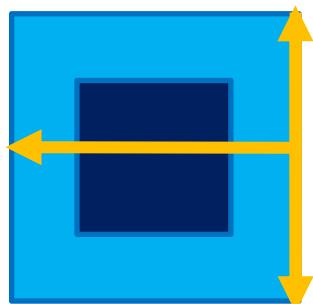
transform-origin: 100px 200px



transform-origin: 100% 50%

transform-origin: right

transform-origin: 200px 100px



原始大小

200px*200px

放大后

400px*400px

扩展知识：在2D变换中，变换中心点的 z-axis 值可以不写，默认为0

- 语法介绍

- skew(x-angle,y-angle) 定义 2D 倾斜转换，沿着 X 和 Y 轴。
- skewX(angle) 定义 2D 倾斜转换，沿着 X 轴。
- skewY(angle) 定义 2D 倾斜转换，沿着 Y 轴。

- 如何使用

- transform: skew(x-angle,y-angle)
- transform: skewX(angle)
- transform: skewY(angle)

- 参数说明

- angle 表示旋转的角度，单位是deg。



```
.box1{ transform: skew(15deg, 15deg) }  
.box2{ transform: skew(-15deg, -15deg) }  
.box3{ transform: skewX(15deg) }  
.box4{ transform: skewX(-15deg) }  
.box5{ transform: skewY(15deg) }  
.box6{ transform: skewY(-15deg) }
```

1. 只能对块元素设置缩放
2. 可以设置在元素缩放的中心点。
3. 不会影响兄弟元素的位置。
4. 如果父元素未设置大小，不会影响父元素的大小。
5. 元素倾斜后，其内容及子元素的都会发生倾斜，可以对内容或子元素设置“反向”倾斜解决。

- 使用场景
 - 需要使用菱形效果
- 实际运用
 - 个性化菜单
 - 特殊的样式

可能需要的配色：
蔚蓝 #14B1E6
蓝黑 #131639
群青 #2957A5

- 语法介绍
 - 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。
- 如何使用
 - transform: translate(x,y)
 - transform: translateX(n)
 - transform: translateY(n)
- 参数说明
 - x, y, n 表示移动的距离 (length)。例如 200px

扩展知识：

1. 变换中心点对位移函数 translate()不其起作用
2. 位移过大导致父元素出现滚动条，此时需对父元素设置over-flow: hidden

- 使用场景
 - 改变一个元素的位置，但是不影响父元素的大小及兄弟元素的位置。
- 实际使用
 - 动画特效

- 语法介绍
 - `rotate(angle)` 定义 2D 旋转，在参数中规定角度。
- 如何使用
 - `transform: rotate(angle)`
- 参数说明
 - `angle` 表示旋转的角度。例如 `45deg`（顺时针旋转45度），`-45deg`（逆时针旋转45deg）

扩展知识：

1. 改变变换中心点的位置，可以实现不同的旋转效果

- 使用场景
 - 让一个元素旋转时
- 实际使用
 - 表盘
 - 照片墙

- 当两个函数组合使用，例如
 - transform: translateX(200px) scale(2)
 - transform: scale(2) translateX(200px)
- 第一个函数会导致后面的函数“坐标”的变换，因此第一种写法，向左侧移动距离会看来更远。



translateX(200px)
scale(2)



scale(2)
translateX(200px)

200px的参照坐标轴



初始元素



变化后的元素



1. 元素变形，不会影响相邻兄弟元素的位置。
2. 元素变形，如果父元素没有指定高度，不会导致父元素高度的变化。
3. 元素变形，只有在块元素中才生效。
4. 元素变形，元素的文本及其后代元素都会变形。





transform 3D

讲师：许井龙

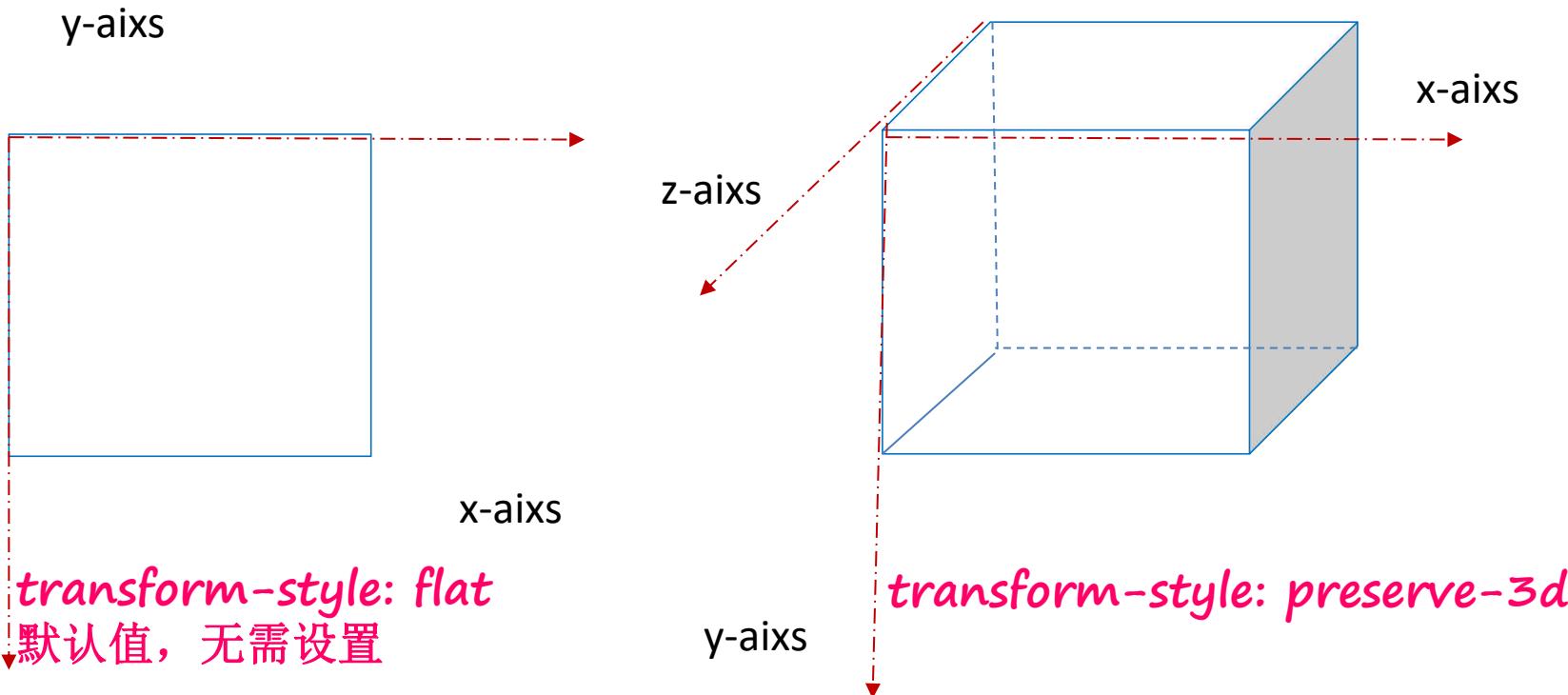
微信：ngsteel

2016年第一版

1. 开启3D空间
2. 景深及景深中心点
3. 3D变换中心点
4. 定义3D元素背面是否可见

- **transform-style**

- flat (默认值) 2D
- preserve-3d 3D



- 父元素开启3D后，子元素设置3D变换函数。就可以看到3D效果。例如
 - 3D旋转 `rotateX(angle)` 或者 `rotateY (angle)`
 - 3D位移 `translateX (length)`

设置变换中心点，看到的效果更明显

你的视角：俯视屏幕

屏幕内



屏幕外

未变换的2D块
元素

开启3D空间的
元素

这是电脑屏幕

你的视角：俯视屏幕

屏幕内

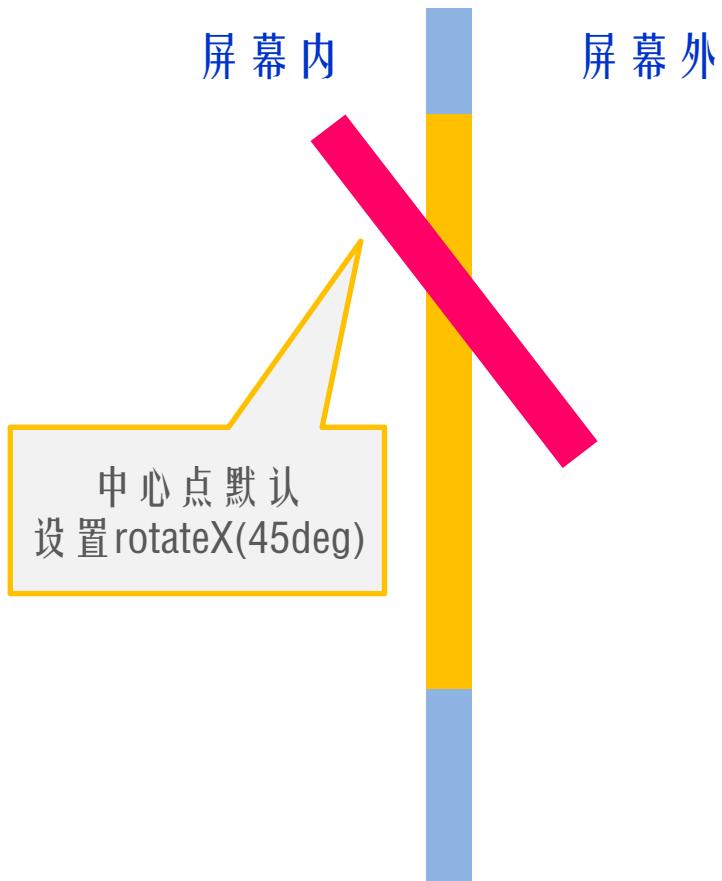


屏幕外



中心点默认
设置 `rotateX(45deg)`

你的视角：屏幕左侧观看



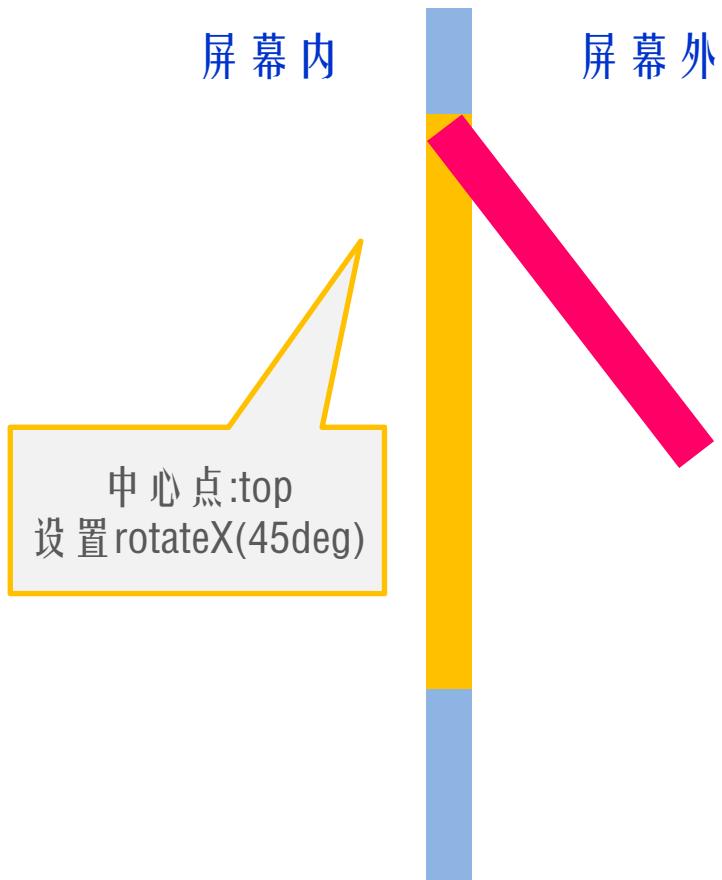
屏幕内

屏幕外



中心点:top
设置rotateX(45deg)

你的视角：屏幕左侧观看



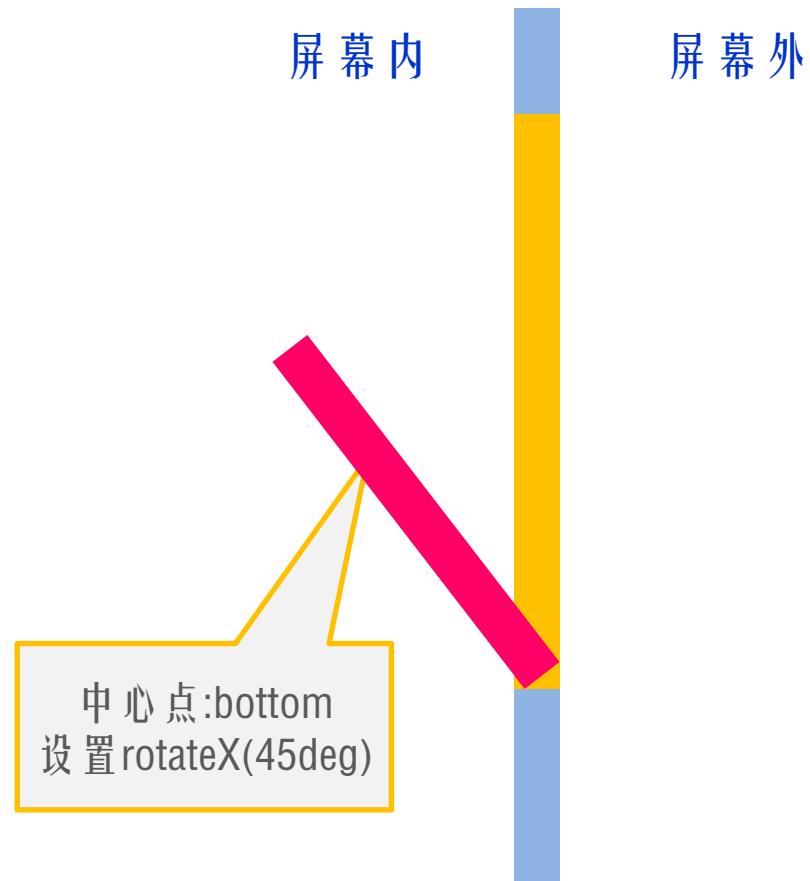
屏幕内



屏幕外

中 心 点 :bottom
设 置 rotateX(45deg)

你的视角：屏幕左侧观看

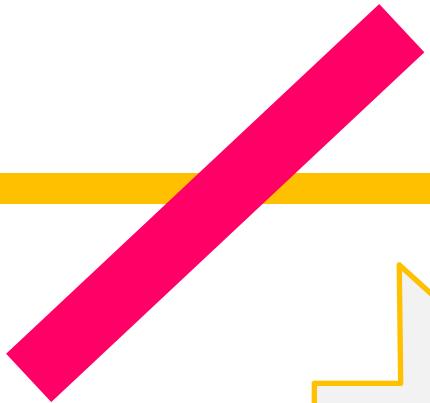


- 当元素设置了rotateX (angle) 此时设置中心点right,或者left是无效的。

你的视角：屏幕上方观看

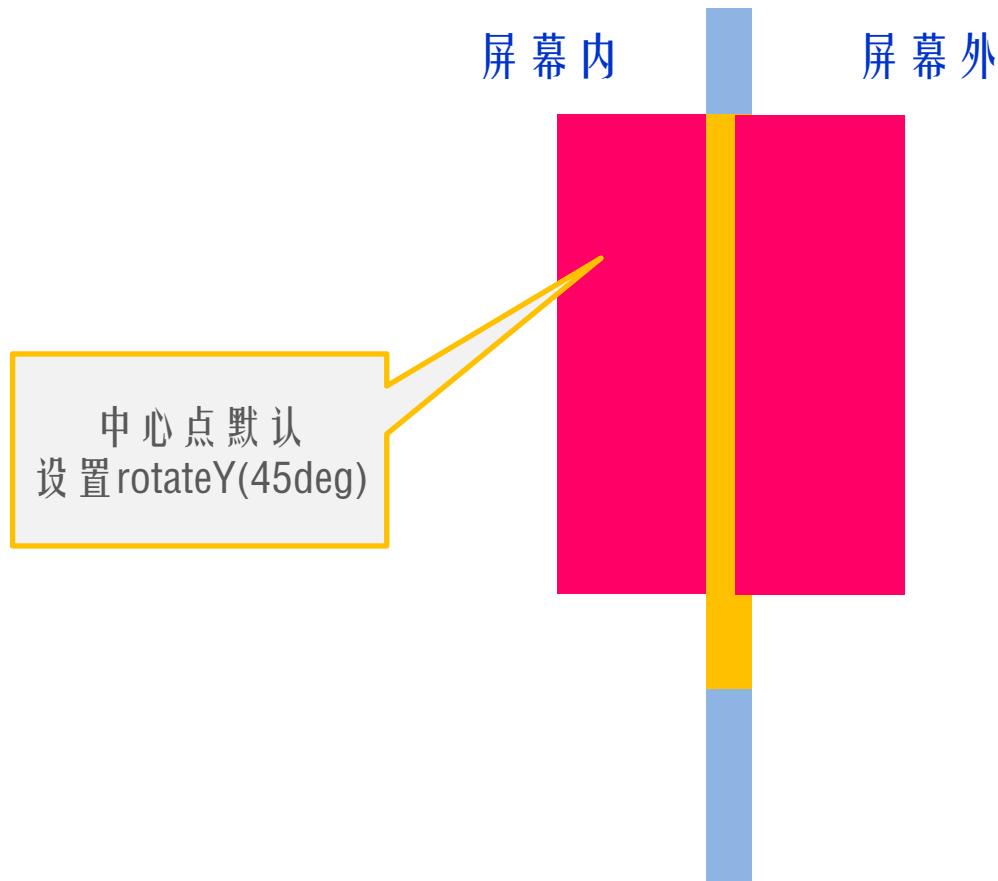
屏幕内

屏幕外



中心点默认
设置rotateY(45deg)

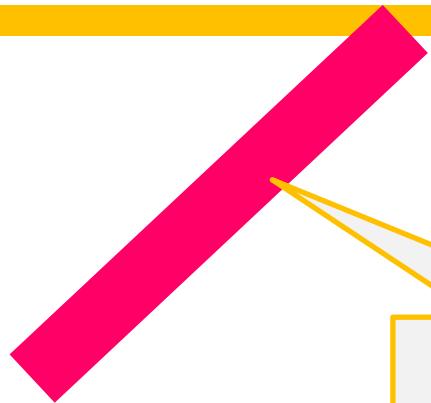
你的视角：屏幕左侧观看



你的视角：屏幕上方观看

屏幕内

屏幕外

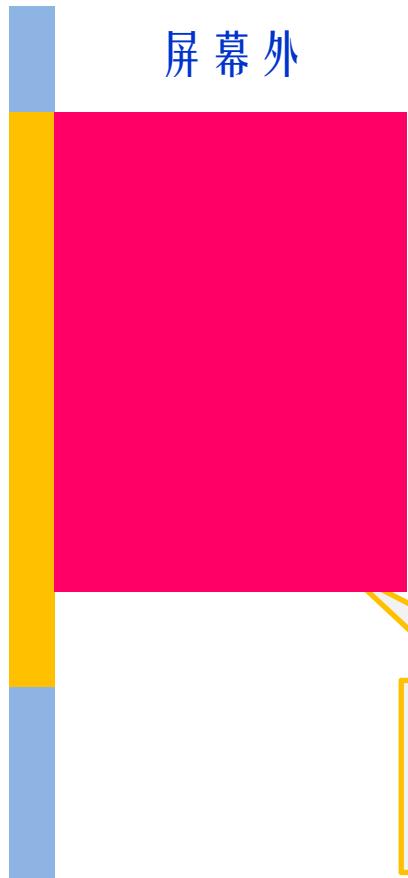


中心点: right
设置rotateY(45deg)

你的视角：屏幕左侧观看

屏幕内

屏幕外

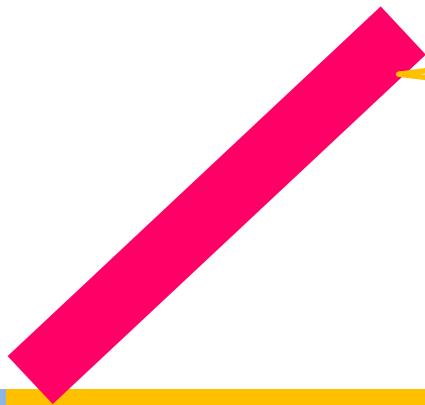


中心点默认
设置rotateY(45deg)

你的视角：屏幕上方观看

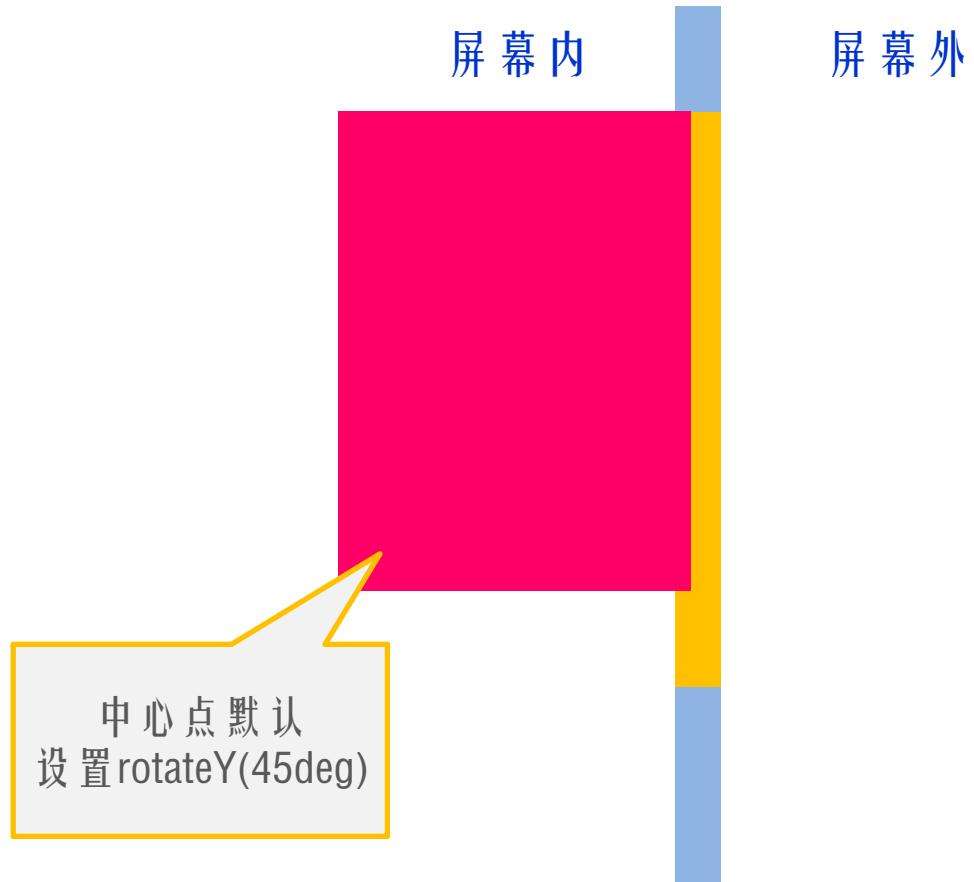
屏幕内

屏幕外



中心点：right
设置 `rotateY(45deg)`

你的视角：屏幕左侧观看



- 当元素设置了rotateY (angle) 此时设置变换中心点top, 或者bottom是无效的。

你的视角：屏幕上方观看

屏幕内



设置translateZ(-200px)



屏幕外



你的视角：屏幕上方观看

屏幕内



屏幕外



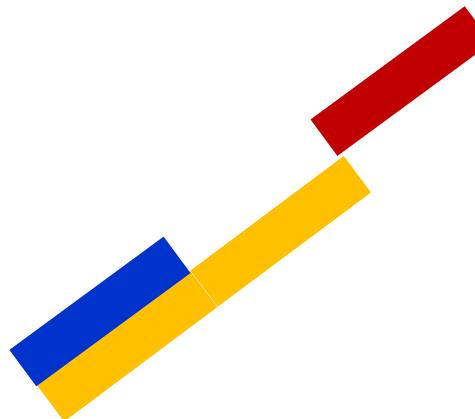
设置translateZ(200px)

- 当两个函数组合使用，例如
 - `transform: translateX(200px) rotateY(45deg)`
 - `transform: rotateY(45deg) translateX(200px)`
- 第一个函数会导致后面的函数“坐标”的变换，因此第一种写法，向左侧移动距离会看来更远。

此刻你的视角：俯视



*`translateX(200px)`
`rotate(45deg)`*



*`rotate(45deg)`
`translateX(200px)`*

200px的参照坐标轴



100px宽高的元素（俯视）

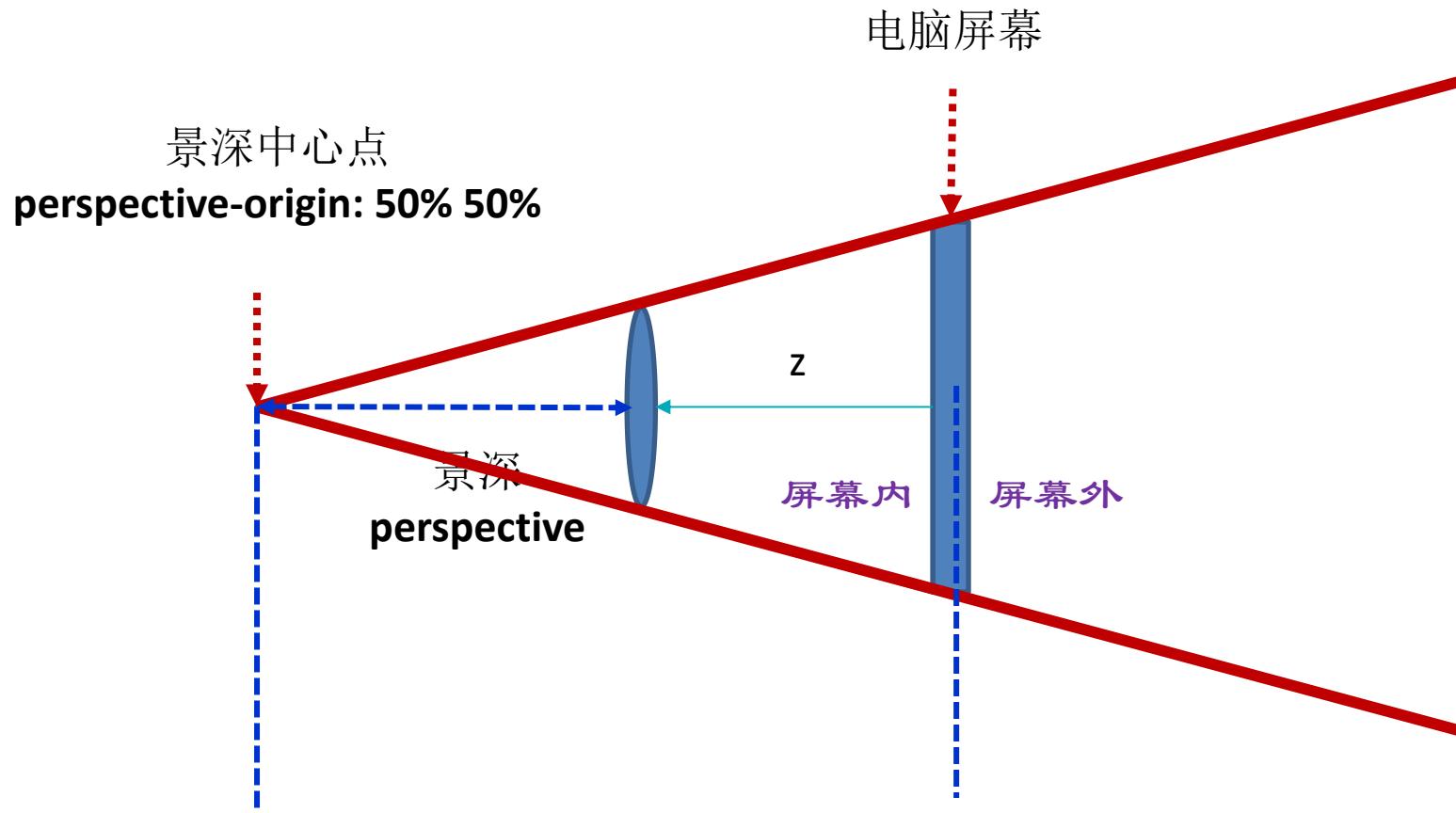


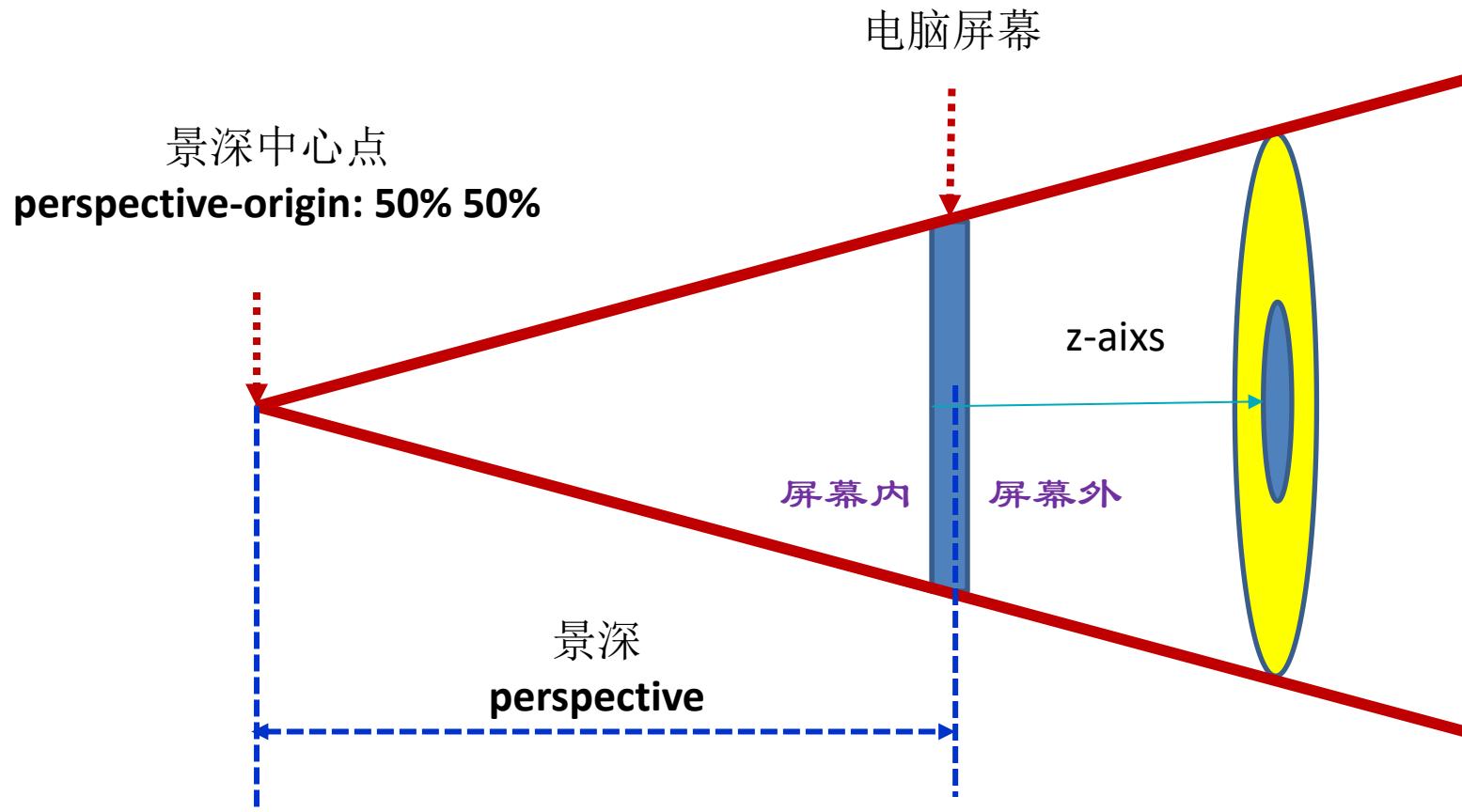
100px宽高的元素变换后的位置



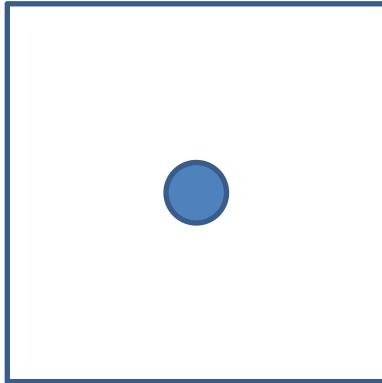
- 无论是2D还是3D位置，设置中心点都是无效的。

- **perspective: *number* | none;**
 - 默认值: 0
 - 元素距离视图的距离, 以像素计。
- **perspective-origin: *x-axis* *y-axis***
 - 默认值: 50% 50%
 - **注意: 景深中心点必须与 perspective 属性一同使用。**

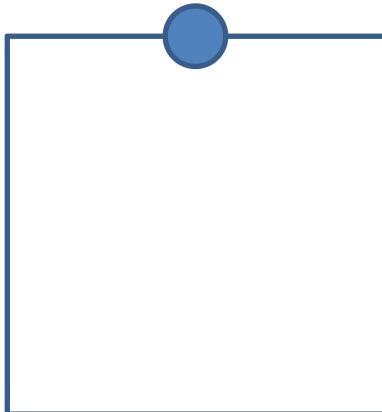




景深及景深中心点

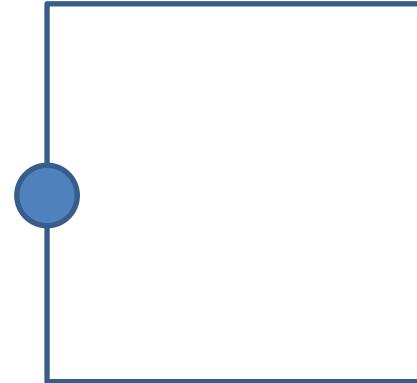


perspective-origin: 50% 50%
perspective-origin: center

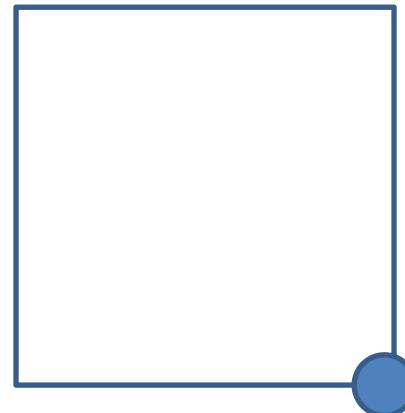


perspective-origin: 50% 0
perspective-origin: top

你的视角：
从屏幕内观察
perspective 的位
置



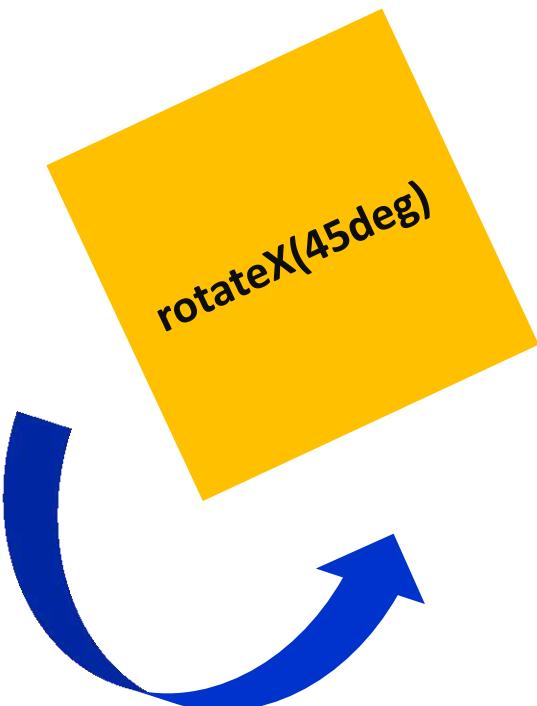
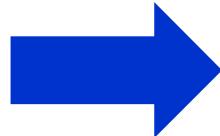
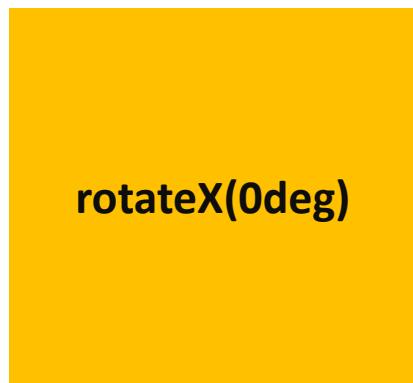
perspective-origin: 0 50%
perspective-origin: left



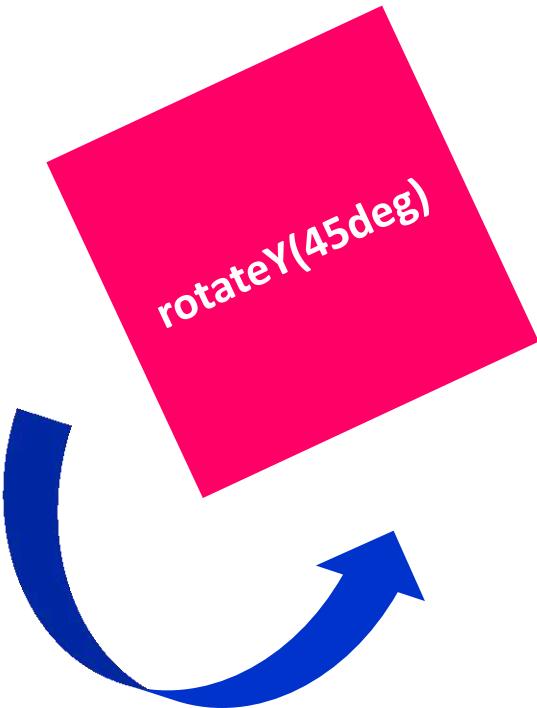
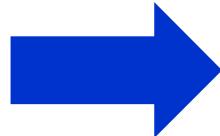
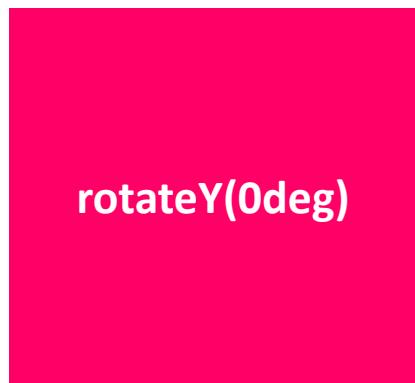
perspective-origin: 100% 100%
perspective-origin: right bottom

- transform-origin: *x-axis y-axis z-axis*;
 - 3D变换的中心点
 - 默认值: 50% 50% 0
 - **注意: z-axis 一定不要使用%！！！ 屏幕内负值，反之为正值。**

你的视角：屏幕 左侧 观看



你的视角：屏幕上上方 俯视 观看



- **backface-visibility: visible|hidden;**
 - visible 背面是可见的。
 - hidden 背面是不可见的。

当一个元素设置了 `rotateY(180deg)`，相当与对元素进行“翻面”，此时如果设置了

`backface-visibility: visible`

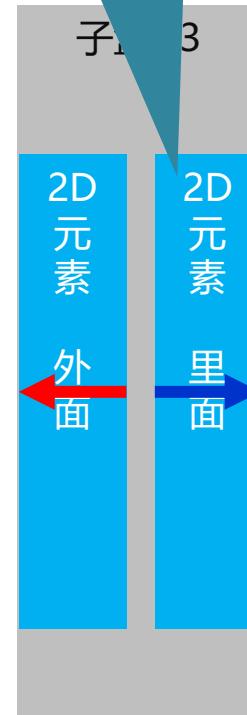
此时该元素不可见。

你的视角：屏幕 左侧 观看

两个2D元素都设
置了绝对定位

3D盒子
旋转180度

里面的2D元素
`rotateY(180deg)`
从屏幕上就可以看到了



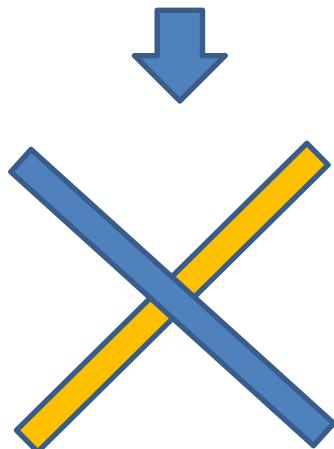
- 利用景深，3D函数组合，我们可以制作3D滚动照片墙



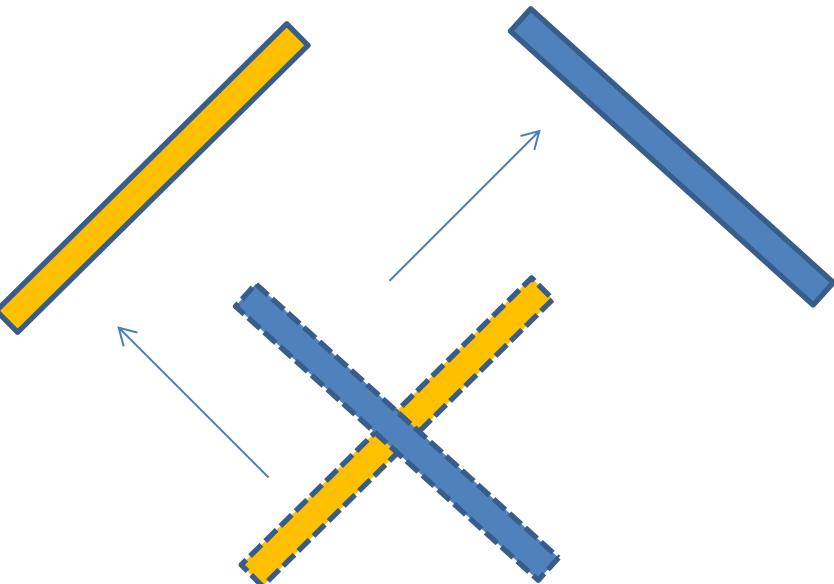
视角：屏幕的上方俯视



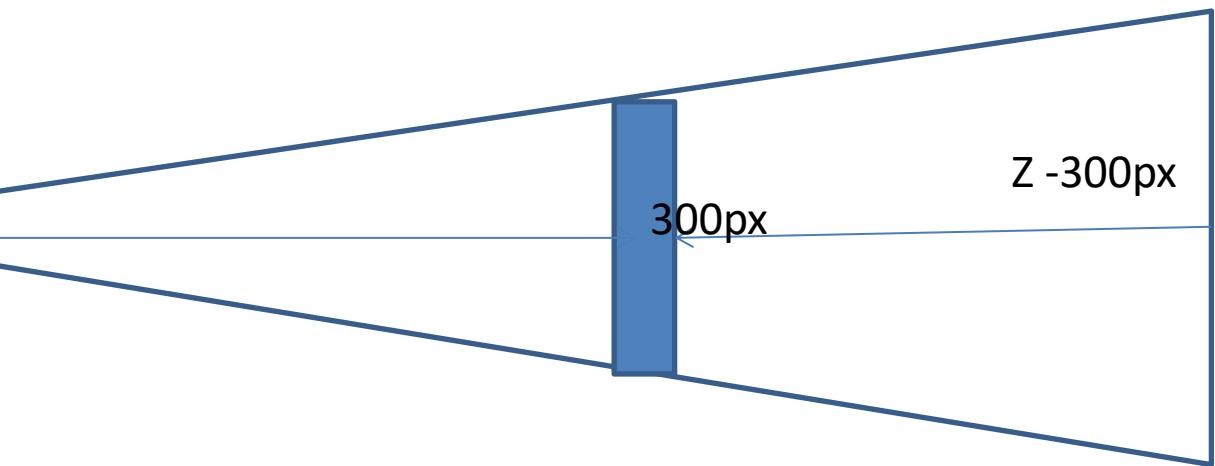
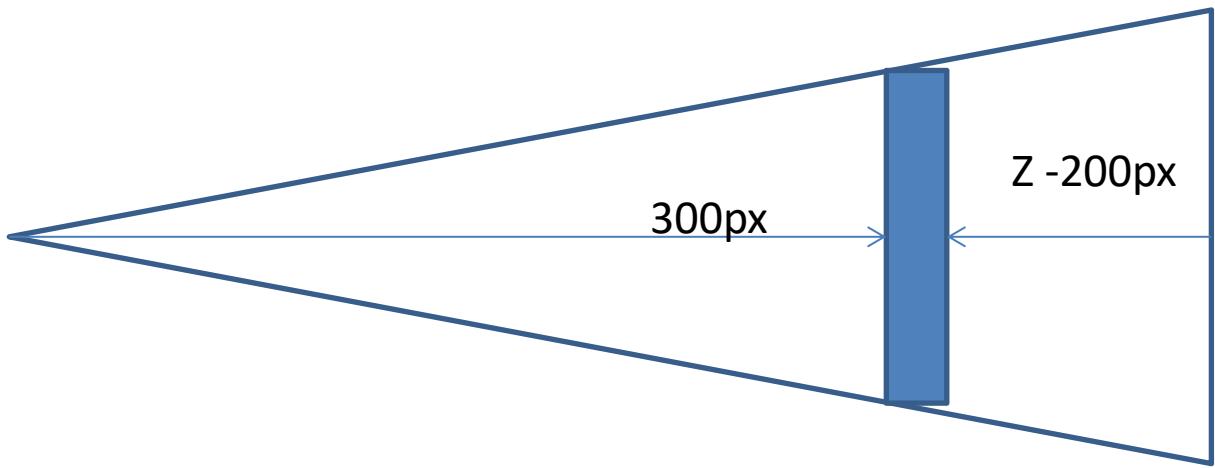
1、所有图片绝对定位

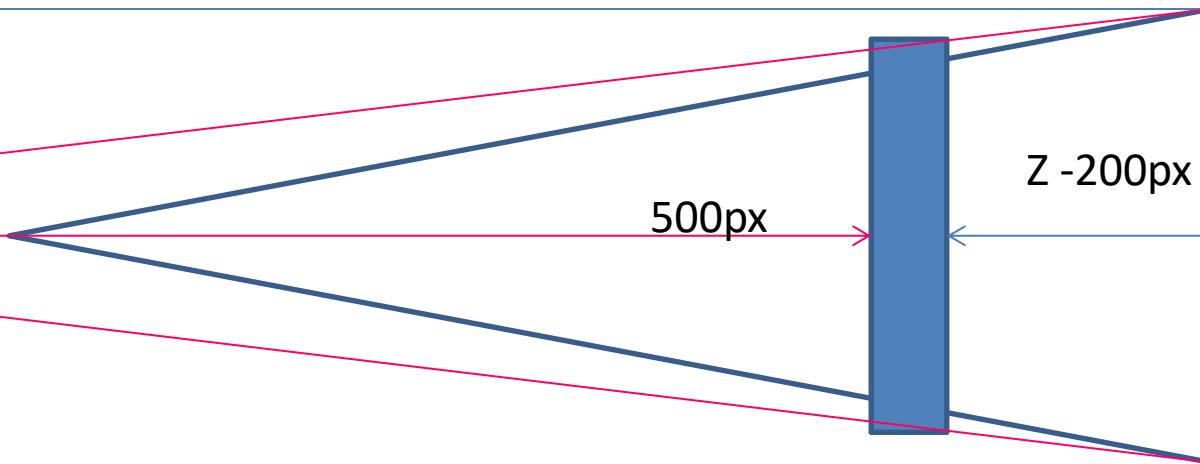
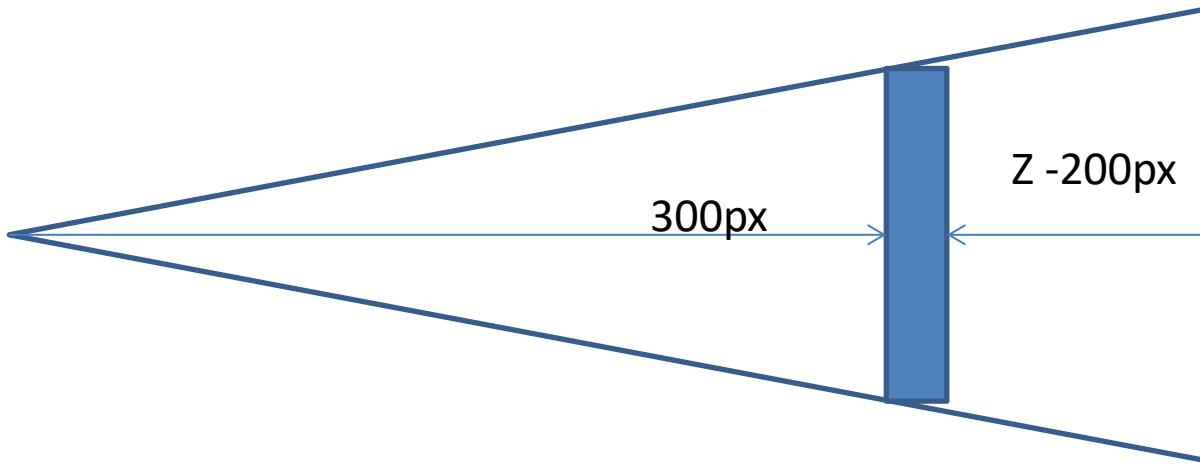


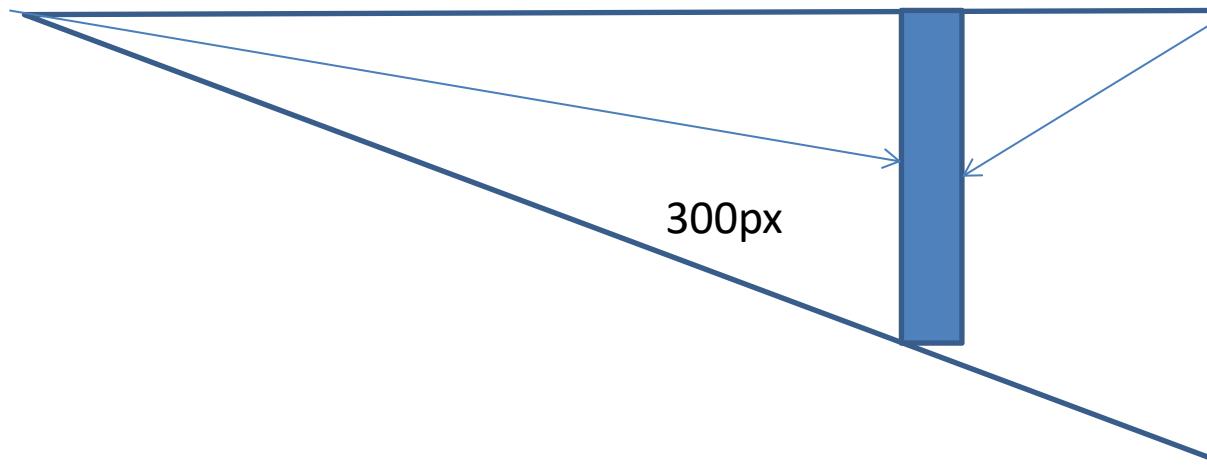
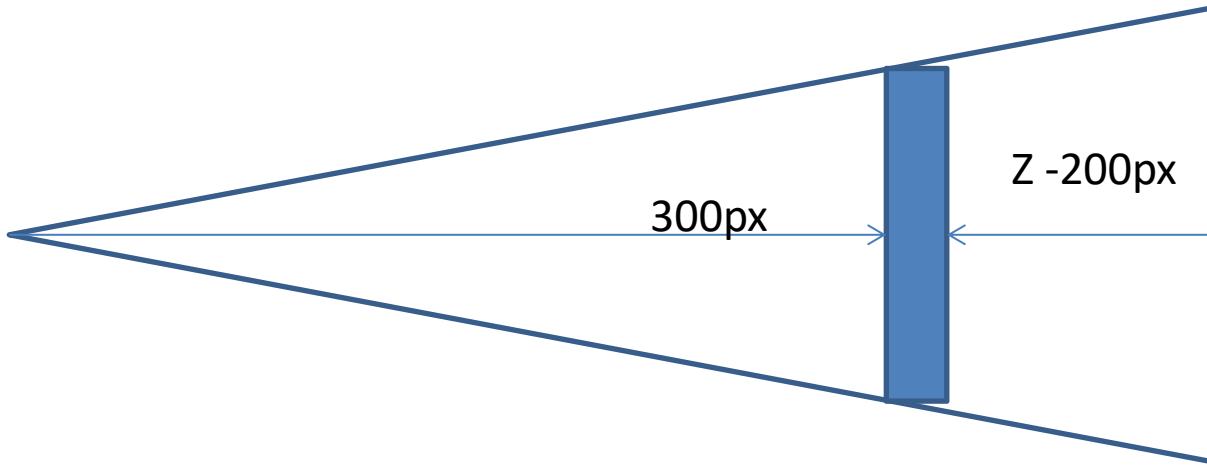
2、依次沿Y轴旋转固定角度
`rotateY(angle)`

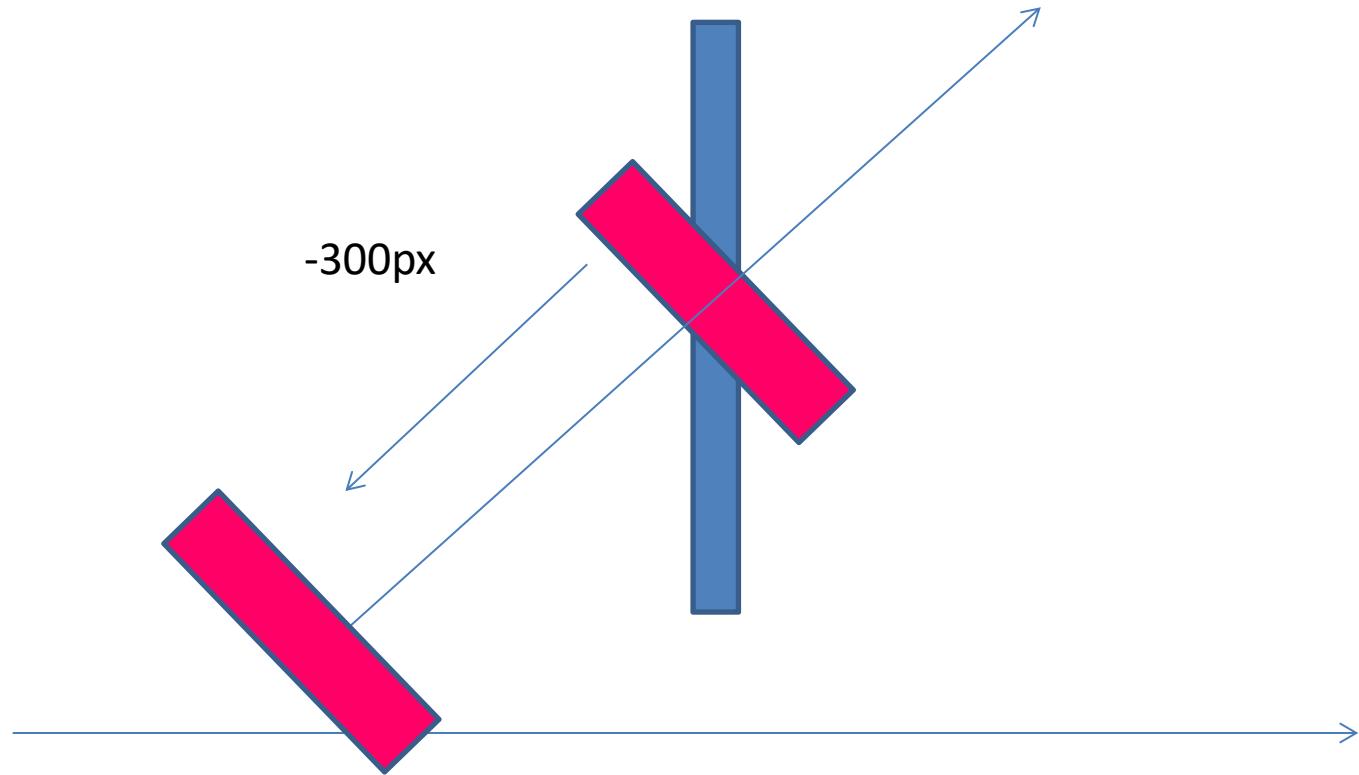


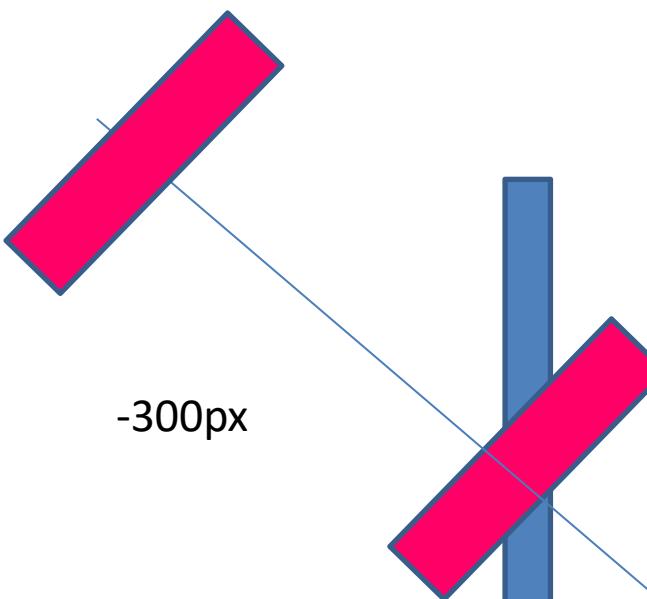
3、依次沿Z轴位移
`translateZ(length)`





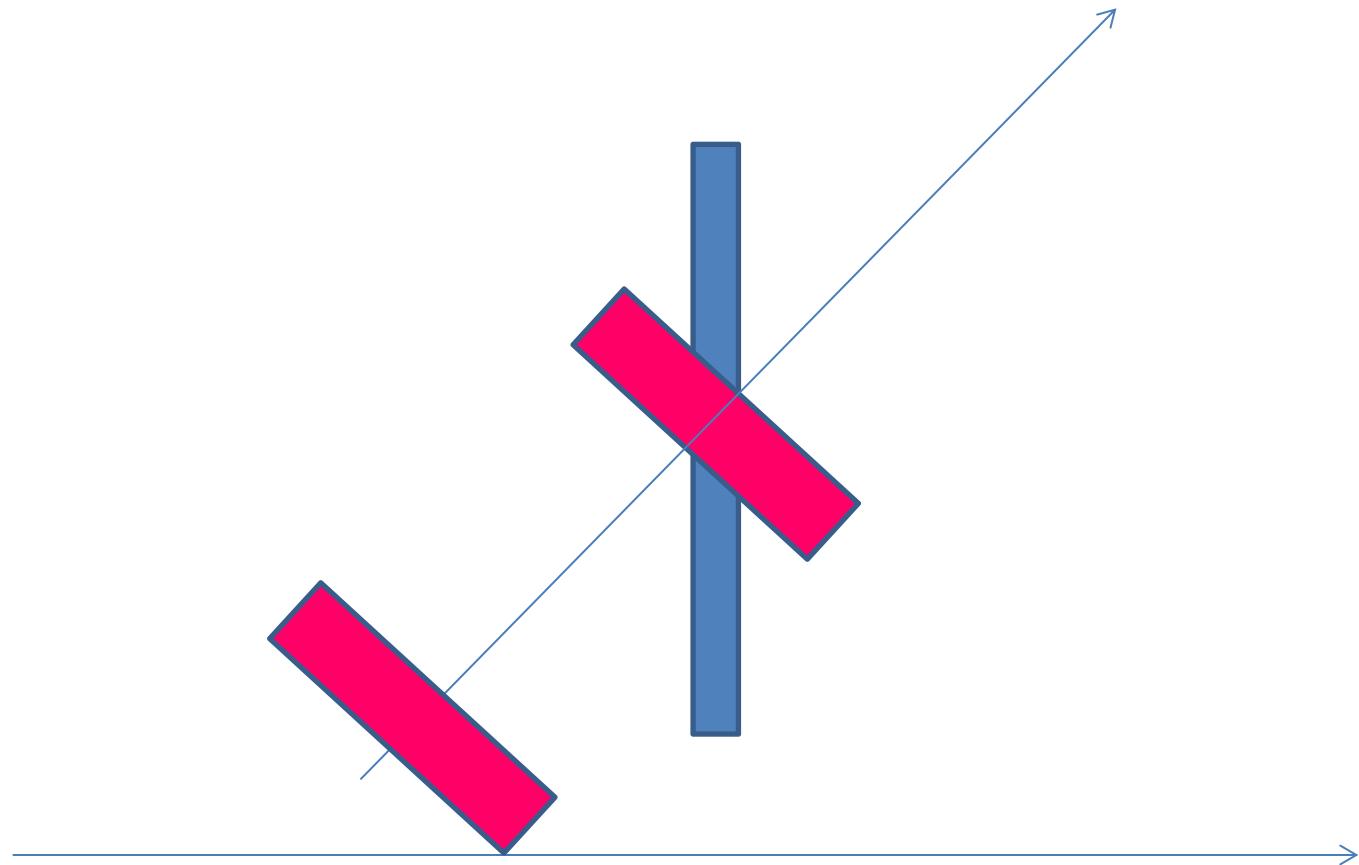


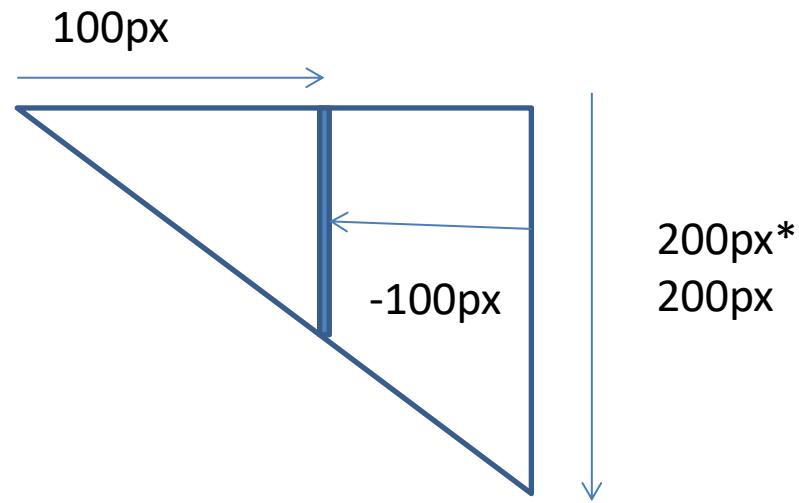




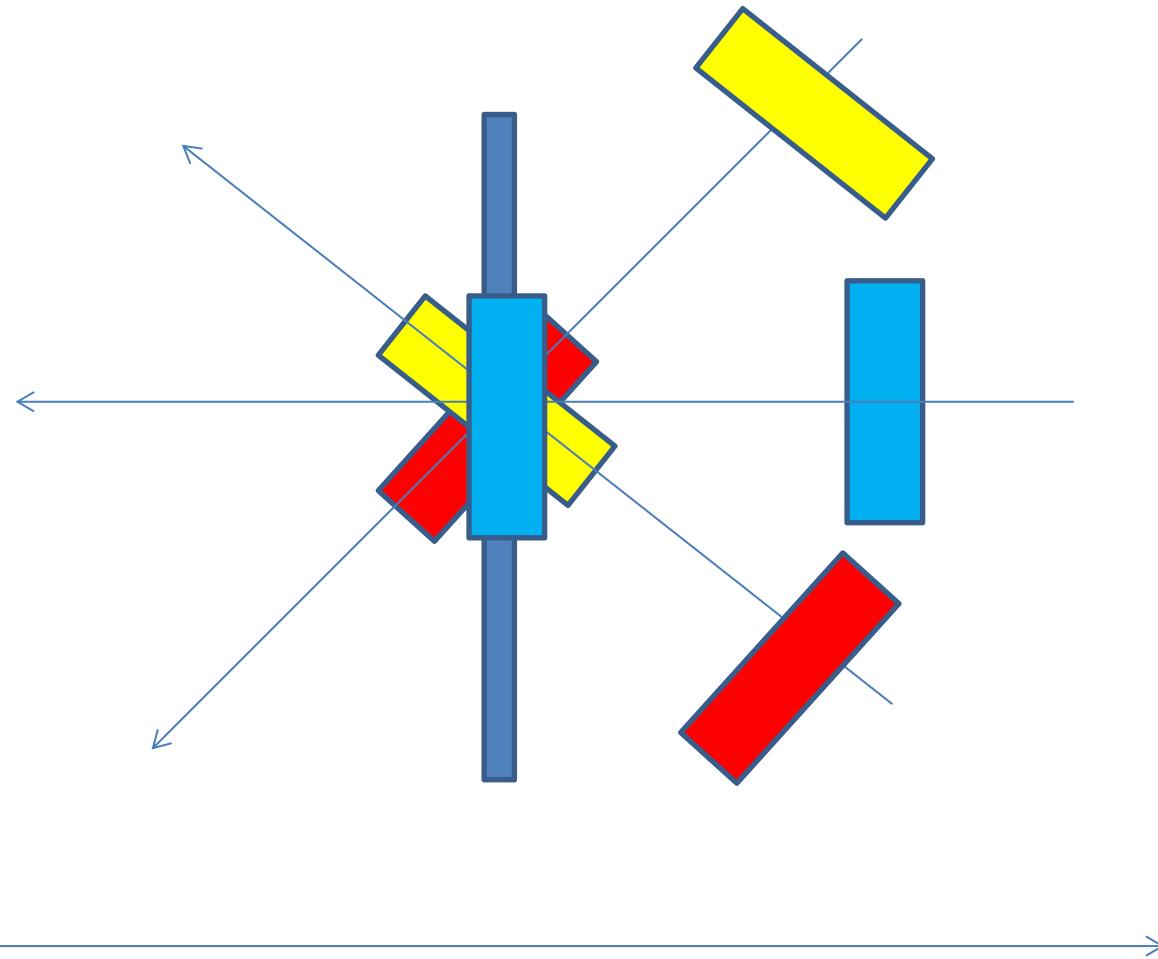
-300px

俯视



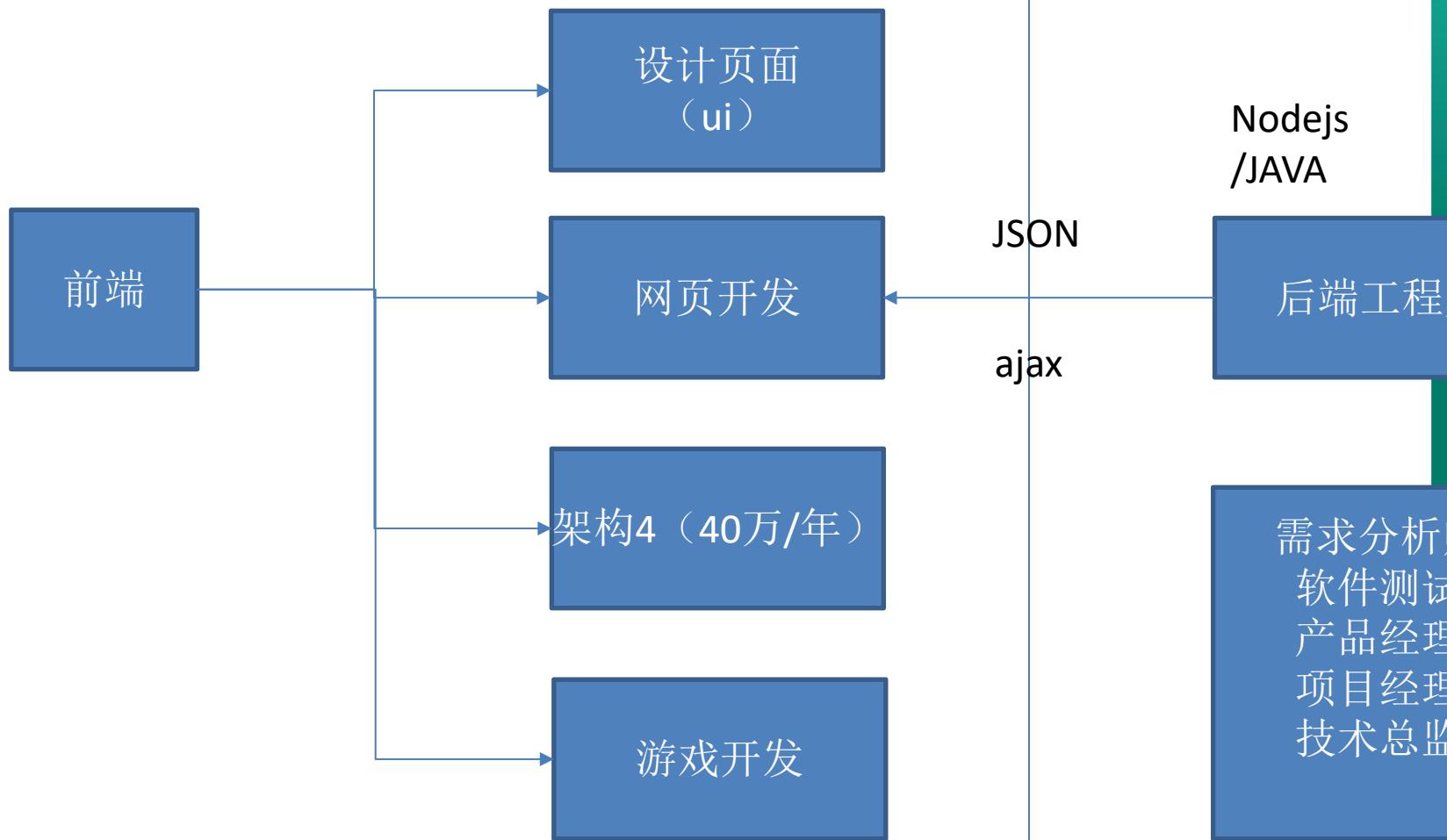


俯视



俯视









CSS3 插件

讲师：许井龙

ngsteel@qq.com

2016年第一版

- 一、浏览器内核及浏览器厂商前缀
- 二、关于prefixfree.js插件

二、浏览器厂商前缀

- -webkit- (Chrome, Safari, 新版Opera.)
 - WebKit/Blink内核
- -moz- (Firefox)
 - Gecko内核
- -o- (旧版Opera)
 - Presto内核
- -ms- (Internet Explorer)
 - Trident / MSHTML内核

WIKI关于浏览器厂商前缀完整介绍：

https://en.wikipedia.org/wiki/CSS_filter#Prefix_filters

二、关于prefixfree.js插件

- **一、功能**
 - 为 `<link>` 或 `<style>` 元素中需要增加前缀的声明自动添加前缀。
- **二、使用限制**
 - 不支持跨域名的样式。
 - 写在元素`style`属性中的没有前缀的样式，不能在IE及Firefox3.6及一下浏览器使用。
- **三、如何使用**
 - 只需要通过`<script>`元素引入prefixfree.js即可，**切记一定要在`<style>`和`<link>`元素后引入。**
- **四、浏览器支持**
 - IE9+, Opera 10+, Firefox 3.5+, Safari 4+ , Chrome (PC端) and Safari (移动端) , Android自带浏览器, Chrome , Opera (移动端)





HTML



媒体特性与响应式设计

讲师：许井龙

ngsteel@qq.com

- 一. 响应式布局简介
- 二. meta标签及视口（viewport）
- 三. 媒体类型
- 四. 媒体特性
- 五. IE兼容性（支持CSS3 Media Queries）
- 六. rem
- 七. 渐进增强和优雅降级
- 八. 浏览器使用情况（百度统计）



随着科技的进步，网页设备终端也越来越多样化。例如智能手机，平板电脑等。如果每个终端都设计一个网页显然不切实际。但是我们**至少要确保不同屏幕尺寸和不同设备上网页看起来是一模一样的。**

- **响应式布局不是新的技术**
 - Ethan Marcotte 在 A List Apart发表了一篇开创性的文章，将三种已有的开发技术（弹性网格布局、弹性图片、媒体和媒体查询）整合起来，并将其命名为RWD（Responsive Web Design，响应式设计）
- **响应式布局的特点**
 1. 网站必须灵活建立网格基础。
 2. 引用到网站的图片必须是可伸缩性的。
 3. 不同设备，需要在Media Query上设置不同的样式

响应式网站赏析

1. 国外 - <http://colly.com/>
2. 国内 - <https://www.wunderlist.com>
3. 国内 - <http://www.micourse.net/>

- 1、**流体网格**：网格大小可以随着屏幕尺寸大小做出相对应的比例缩放。
- 2、**弹性图片**：不给图片设定固定尺寸，根据流体网格自动缩放。一条代码即可搞定：`img{max-width: 100%}`
- 3、**媒体查询 (Media Query)**：响应式设计的灵魂，可以根据设备尺寸，匹配不同的样式。
- 4、**屏幕分辨率**：利用媒体查询对不同分辨率手机设置不同的样式。
- 5、**主要断点**：WEB开发中的新名词，在响应式设计中尤为重要。简单说就是设备的临界点。利用媒体查询针对不同尺寸终端设备设置不同样式。

- **响应式布局技巧：首先要让页面布局尽量简单，实现简单布局的技巧。**
 - 尽量少用无关紧要的div.
 - 不要使用内联元素（没法设置宽度）
 - 尽量少用JS或者flash
 - 丢弃没用的绝对定位或者相对定位
 - 摒弃任何冗余结构
- **哪些方法可以帮助我们更好的实现响应式设计**
 - 使用HTML5 Doctype和相关指南
 - 重置好样式（reset.css）
 - 一个简单的有语义的核心布局
 - 不要过分依赖现代技巧来实现，比如CSS3特效或者JS脚本

- 如何检查HTML结构是“简单干净”的?
 - 禁掉所有的CSS样式，如果内容排列有序，方便阅读，那么这个结构就不会差到哪里去。

1. 视觉视口: **用户正在看到的网站的区域**, 一般视觉视口和布局视口一致。并且它的CSS像素的数量会随着用户缩放而改变。
 - ◆ 可通过 `window.innerWidth/innerHeight` 获取;
2. 布局视口: 不再与移动端浏览器相关联, 完全是独立的。实际上**布局视口的宽度要比屏幕高出很多**。了容纳为桌面浏览器设计的网站, 移动设备默认的布局视口宽度远大于屏幕的宽度, 设置为 **980px** 或 1024px (也可能是其它值, 这个是由设备自己决定的), 但带来的后果就是浏览器会出现横向滚动条, 因为浏览器可视区域的宽度是比这个默认的viewport的宽度要小的
 - ◆ 可通过 `document.documentElement.clientWidth/clientHeight` 获取;
3. 理想视口: 布局视口的默认宽度并不是一个理想的宽度。显然用户希望在进入页面时可以不需要缩放就可以有一个理想的浏览和阅读尺寸。理想视口仍是为移动端设备准备的。只有手动添加meta视口标签方才生效。如果没有meta视口标签, 那么布局将会维持它的默认宽度。
 - ◆ 让网页的宽度自动适应手机屏幕的宽度。

二、meta标签及视口 (viewport)

- <meta name="viewport" content=" " >告诉浏览器如何处理网页。该标签主要作用让移动设备在浏览网页时进行优化，并且可以自定义界面可视区域的尺寸与缩放级别。
- viewport: 智能手机使用了一个比实际屏幕大的虚拟可视区域 (viewport)，主要的目的是让页面在智能手机阅读时不会因为实际可视区域而变形。

- <meta name="viewport" content="">
 - width : 可视区域宽度，可以是一个具体的数字也可以是device-width。**可以让网页的宽度自动适应手机屏幕的宽度**
 - height : 可视区域高度，可以是一个具体的数字也可以是device-height。
 - initial-scale: 页面首次显示时，可视区域的缩放比例，取值1.0页面按实际比例显示，无任何缩放。
 - minimum-scale: 可视区域最小缩放级别，取值1.0禁止用户缩放至实际尺寸以下。
 - maximum-scale: 可视区域最大缩放级别，取值1.0禁止用户放大至实际尺寸以上。
 - user-scalable: 用户是否可以对页面进行缩放。yes可以缩放，no禁止缩放。

移动端的网页需要设置如下meta

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
```

媒体类型定义

值	描述
<i>all</i>	用于所有设备
<i>print</i>	用于打印机和打印预览
<i>screen</i>	用于电脑屏幕，平板电脑，智能手机等。
<i>speech</i>	应用于屏幕阅读器等发声设备
<i>aural</i>	已废弃。用于语音和声音合成器
<i>braille</i>	已废弃。应用于盲文触摸式反馈设备
<i>embossed</i>	已废弃。用于打印的盲人印刷设备
<i>handheld</i>	已废弃。用于掌上设备或更小的装置，如PDA和小型电话
<i>projection</i>	已废弃。用于投影设备
<i>tty</i>	已废弃。用于固定的字符网格，如电报、终端设备和对字符有限制的便携设备
<i>tv</i>	已废弃。用于电视和网络电视

媒体类型使用

在打印设备中，使用print.css里定义的样式

```
<link rel="stylesheet" type="text/css" href="print.css" media="print">
```

直接通过@media选择器（Media Query）定义在<style>元素中

```
@media print {  
    /*样式代码 */  
}
```

min-width: 768px



断点
768px

max-width: 1024px

断点
1024px

- 媒体属性 (**Media features**)：大多数媒体属性带有“min-”和“max-”前缀，用于表达“小于等于”和“大于等于”。这避免了使用与HTML和XML冲突的“<”和“>”字符。如果你未向媒体属性指定一个值，并且该特性的实际值不为零，则该表达式被解析为真。

- width**

- max-width**
- min-width**

- orientation**

- landscape 横屏**
- portrait 竖屏**

- Media Query 使用方法
 - @media 媒体类型 and (媒体特性) { 样式 }

```
@media screen and (min-width: 768px) {  
    /** 执行代码 样式 */  
}
```

768px 是**断点**，768px下屏幕尺寸

```
@media screen and (min-width: 640px) and (max-width: 768px) {  
    /** 执行代码 样式 */  
}
```

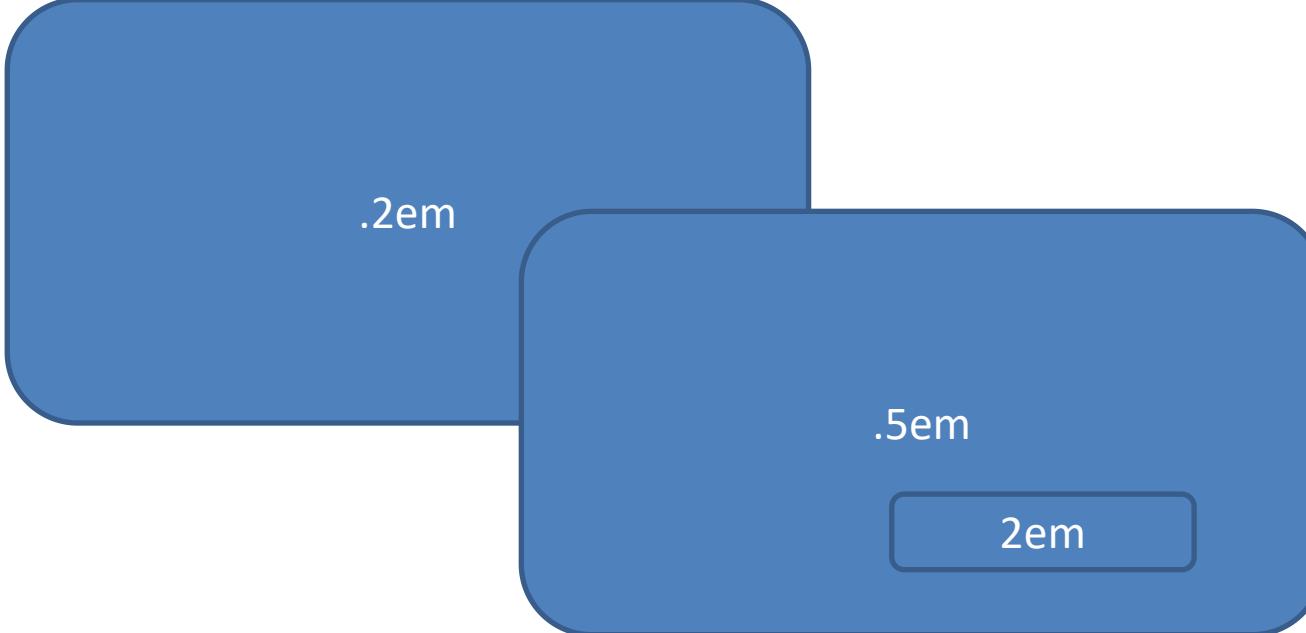
- rem (root em) , 与px , em 一样也是一个单位。不同的是它是基于html元素 ($\text{html} = \text{root element}$) 计算。
- 该单位常与媒体查询一起使用。
- 优点是：只要改变html的font-size大小，所有基于该元素计算的元素相关尺寸随之等比改变。

- rem 缺点
 - $1\text{rem} = 16\text{px}$ 需要进行换算，非常麻烦。
 - 解决办法：html{ font-size: 62.5%}，此时可以使用1.2rem表示12px，计算比较方便。但是致命的问题随之而来，Chrome浏览器最小字体是12px，如果设置了11px及一下，仍显示12px。
 - 目前普遍被接收的解决办法：html{ font-size: 625%}。但是这种方法使用不是很方便。

- rem 缺点

- $1\text{rem} = 16\text{px}$ 需要进行换算，非常麻烦。
 - 解决办法：html{ font-size: 62.5%}，此时可以使用1.2rem表示12px，计算比较方便。但是致命的问题随之而来，Chrome浏览器最小字体是12px，如果设置了11px及一下，仍显示12px。
 - 目前普遍被接收的解决办法：html{ font-size: 625%}。但是这种方法使用不是很方便。

- px: 在PC端使用, 移动端谨慎使用。
- em: 避免嵌套使用, **基于自身字体的大小**。
- %: 避免嵌套使用, **基于父元素**。
- rem: **基于html元素字体大小**。注意Chrome不支持12px以下像素



.2em

.5em

2em

- border-box
- content-box

- **渐进增强**: 针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。
- **优雅降级**: 一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

- 第三方插件：
 - <https://github.com/scottjehl/Respond>
- 注意事项：
 - 1、需要在服务器上运行。
 - 2、包含媒体查询的 css文件需采用外链形式。
 - 3、**头部引用的respond.js 需置于css 文件之后。**

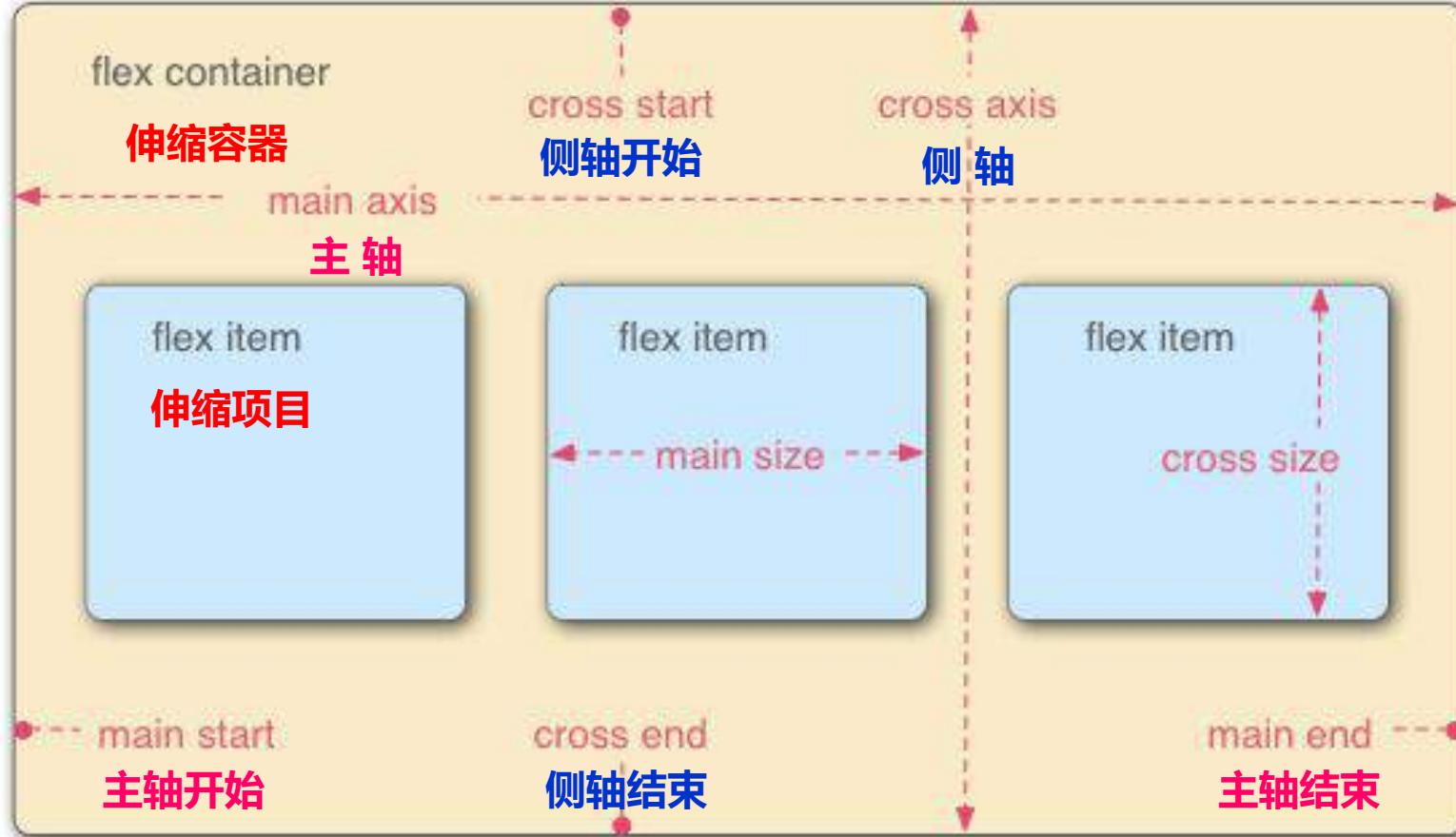


CSS3 flex

讲师：许井龙

微信 : ngsteel

- 伸缩盒模型 flex 是CSS3中快速布局的利器。



- 为元素设置
 - `display: flex`
 - 或者 `display: inline-flex`
- 该元素即成为**伸缩容器 (flex container)**，
 - 设置为 `flex`，该元素会独占一行
 - 设置为 `inline-flex`，可以与其他 `inline` 元素在同一行。
- 此时伸缩容器的子元素自动升级为**伸缩项目 (flex item)**，伸缩项目的特点如下，
 1. 伸缩项目默认在一行排列。
 2. 自动升级为块元素。
 3. 所有伸缩项目默认在主轴的 `start` 处排列。
 4. 伸缩项目也可以再次设置为 `flex`，即 `flex` 可以互相嵌套。

- inline-flex 行内对齐特点

- 伸缩容器中有文本内容，基于第一个文本的基线对齐。
- 伸缩容器中没有有文本内容，也没有子元素，伸缩容器底边位于一行的基线处。
- 伸缩容器中没有有文本内容，第一个子元素没有，第一个元素底边位于一行的基线处。

伸缩容器有足够的空间
伸缩项目排列

- 当伸缩容器有“足够空间”时，所有伸缩项目在主轴start处排列，如同设置了
 - justify-content: flex-start (默认值)



- 当伸缩容器有“足够空间”时，也可以设置，
 - justify-content: flex-end
- 此时所有的伸缩项目在伸缩容器主轴的end处排列



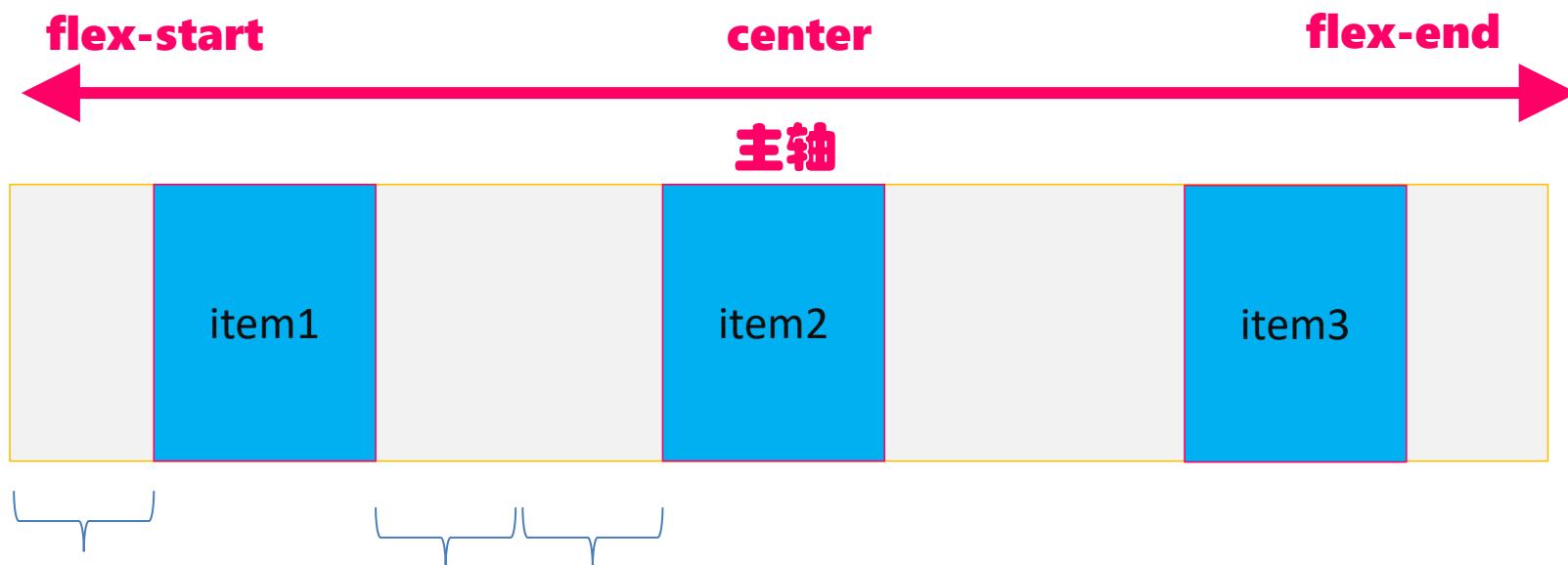
- 当伸缩容器有“足够空间”时，设置
 - justify-content: center
- 所有的伸缩项目位于伸缩容器主轴的中间处



- 当伸缩容器有“足够空间”时，设置
 - justify-content: space-between
- 此时伸缩项目沿着伸缩容器的主轴均匀分布



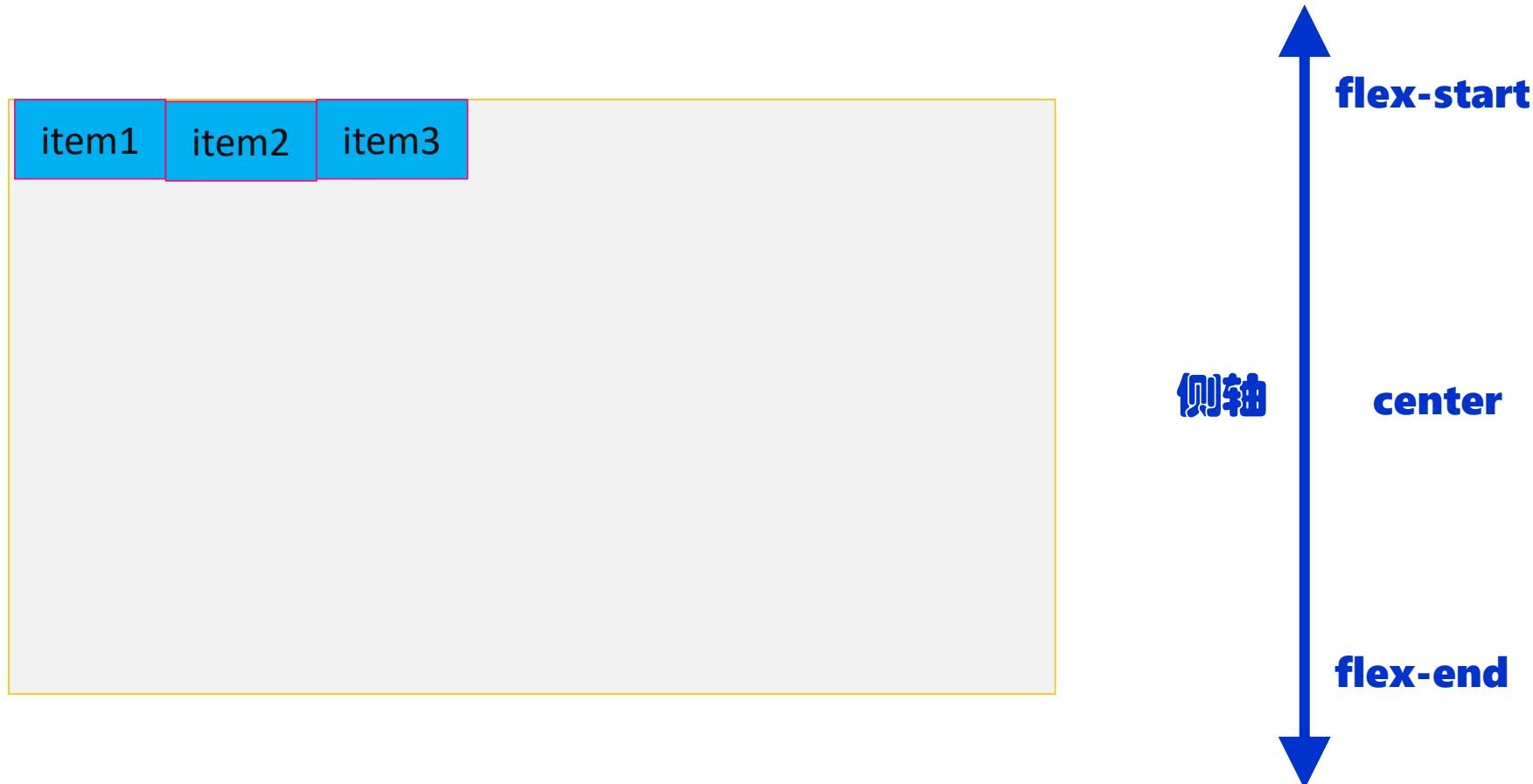
- 当伸缩容器有“足够空间”时，设置
 - justify-content: space-around
- 伸缩项目沿着伸缩容器的主轴均匀分布，但是剩余空间会包裹着每个伸缩项目



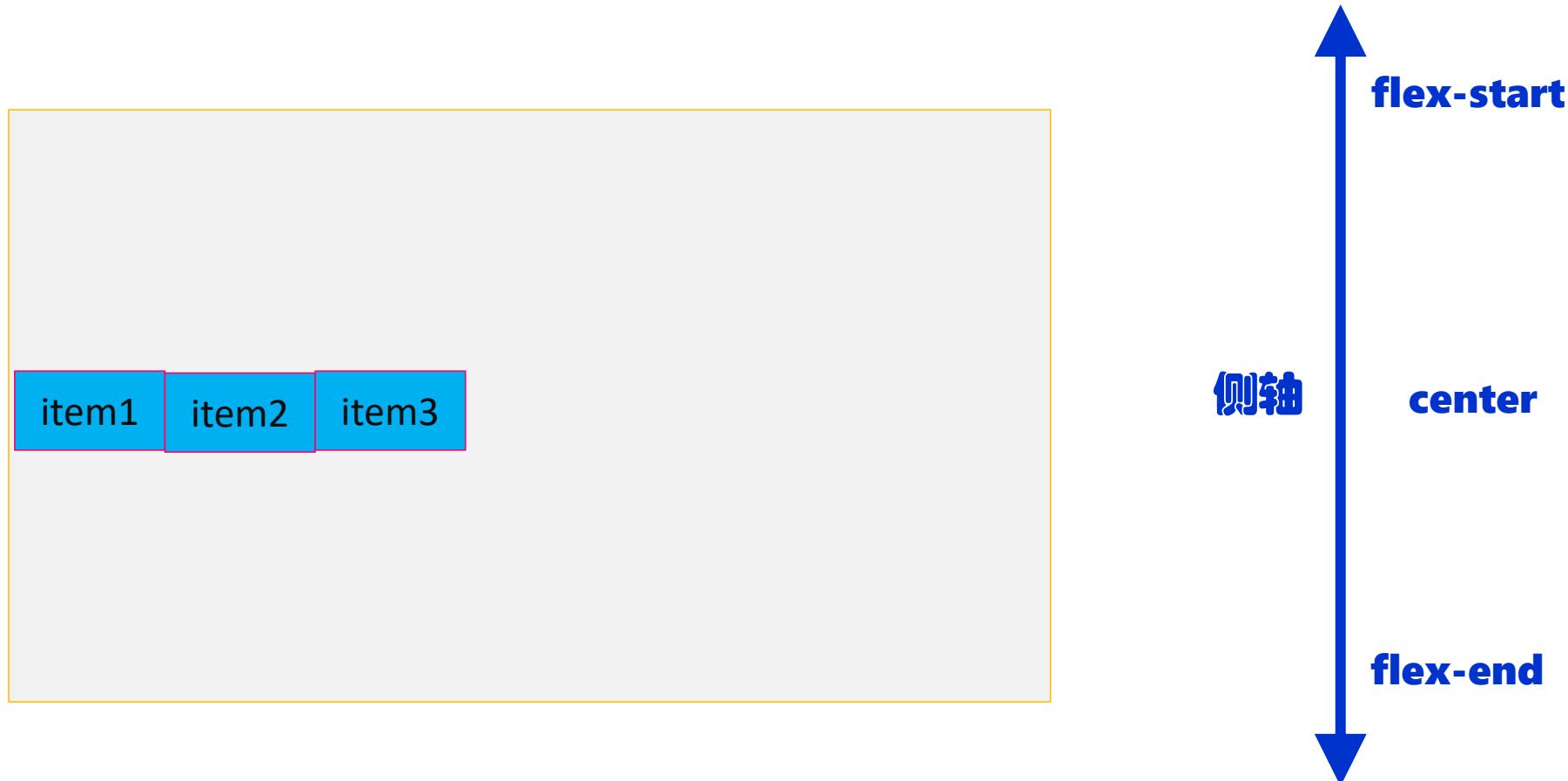
- 当伸缩容器有“足够空间”时，侧轴如同设置了
 - align-items: stretch
- 即每个伸缩项目会沿侧轴被拉伸。



- 当伸缩容器有“足够空间”时，如果设置，
 - align-items: flex-start
- 此时所有伸缩项目位于伸缩容器侧轴start处



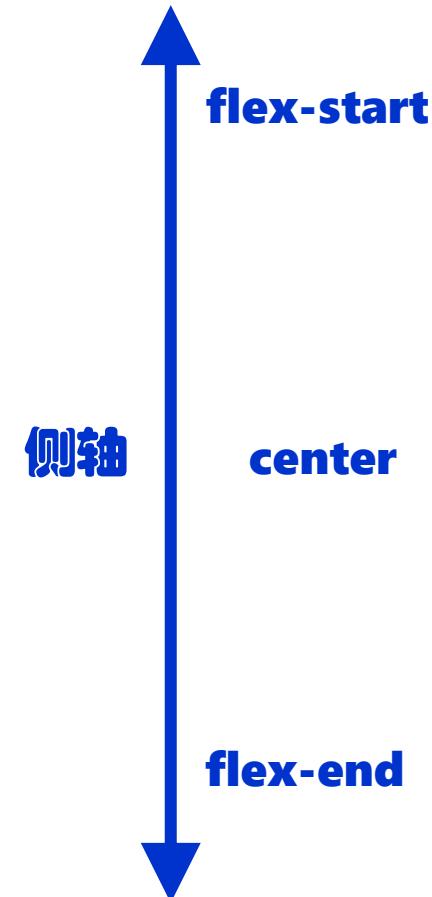
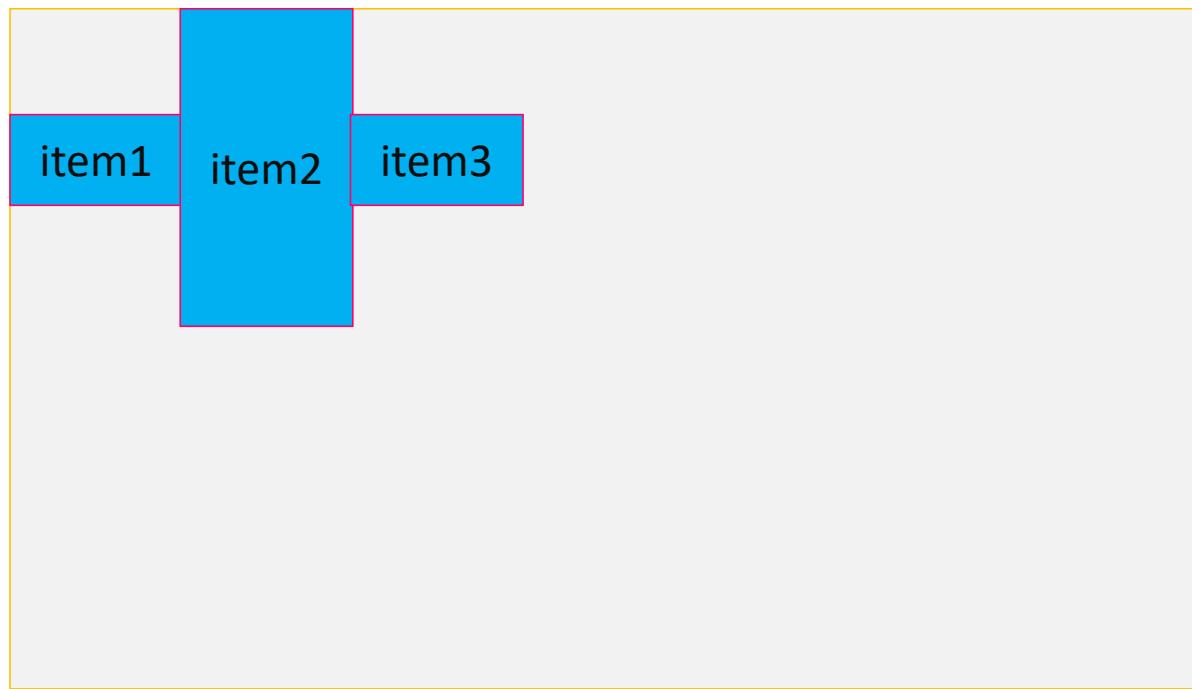
- 当伸缩容器有“足够空间”时，如果设置，
 - align-items: center
- 此时所有伸缩项目位于伸缩容器侧轴中部



- 当伸缩容器有“足够空间”时，如果设置，
 - align-items: flex-end
- 此时所有伸缩项目位于伸缩容器侧轴end处



- 当伸缩容器有“足够空间”时，如果设置
 - align-items: baseline
- 主轴中伸缩项目基线最大的那个伸缩项目的基线作为所有伸缩项目的对齐基线**



伸缩容器空间不足

伸缩容器主轴空间不足

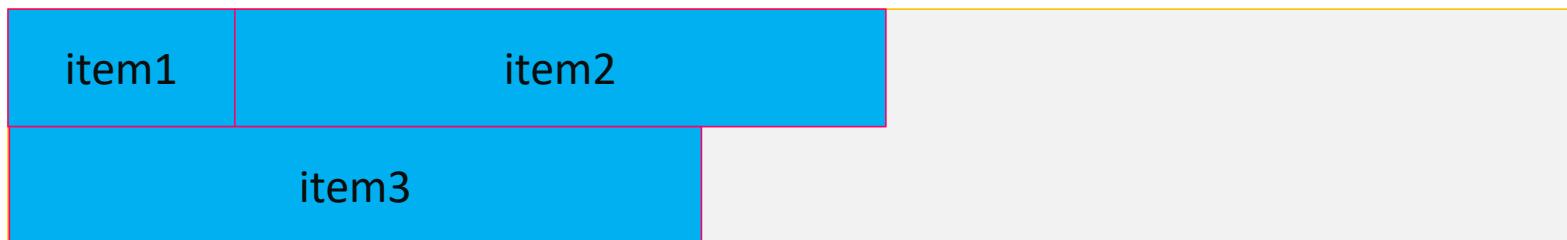
- 当设置伸缩项目大小，且伸缩容器无法在主轴方向上容纳所有的伸缩项目时，**每个伸缩项目都会被按比例被压缩**。
如同在伸缩容器中设置了，
 - `flex-wrap: nowrap` (默认值)



如何压缩，请参考伸缩项目压缩率计算规则

伸缩容器主轴空间不足

- 当伸缩项目设置大小，且伸缩容器无法在主轴方向上容纳所有的伸缩项目时，在伸缩容器中设置，
 - flex-wrap: wrap
- 让伸缩项目自动换行，保持原来的大小。

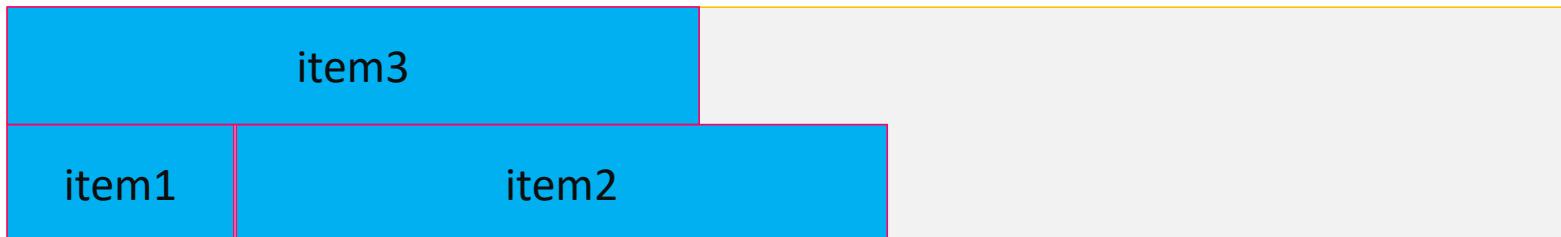


伸缩容器主轴空间不足

- 当伸缩项目设置大小，且伸缩容器无法在主轴方向上容纳所有的伸缩项目时，在伸缩容器中设置，
 - flex-wrap: wrap-reverse
- 让伸缩项目自动换行，保持原来的大小。但此时伸缩项目在主轴flex-start和侧轴flex-end处开始排列。

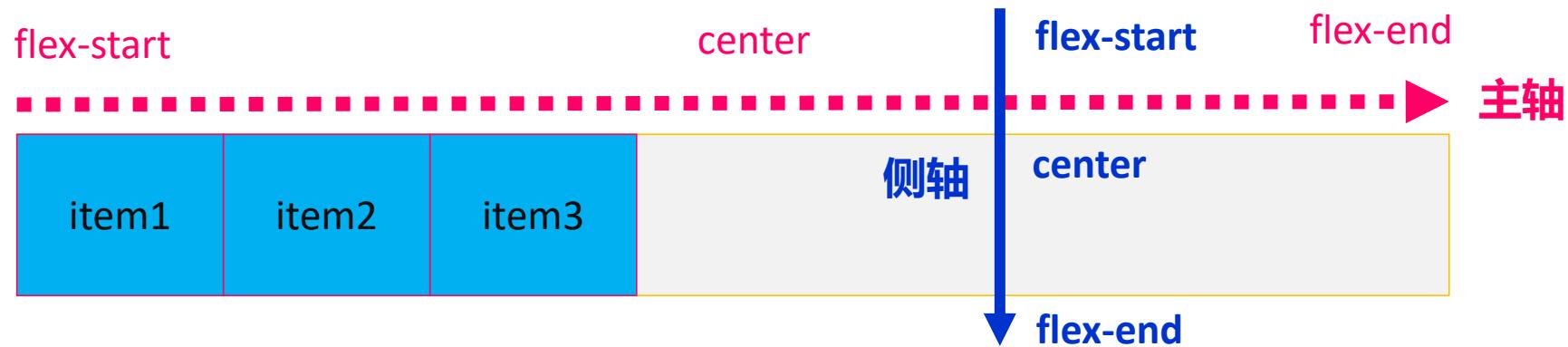


flex-wrap: wrap-reverse;

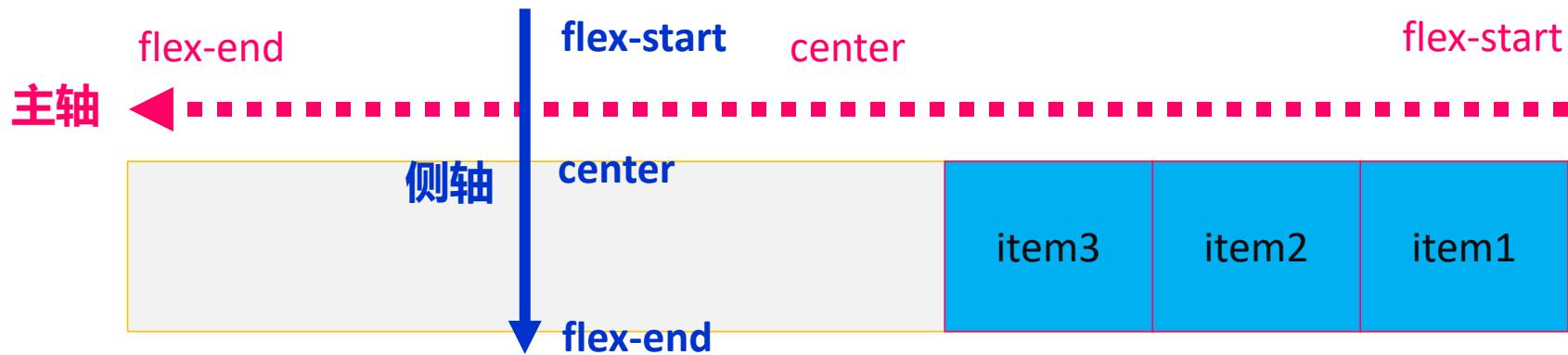


主轴及侧轴相关设置

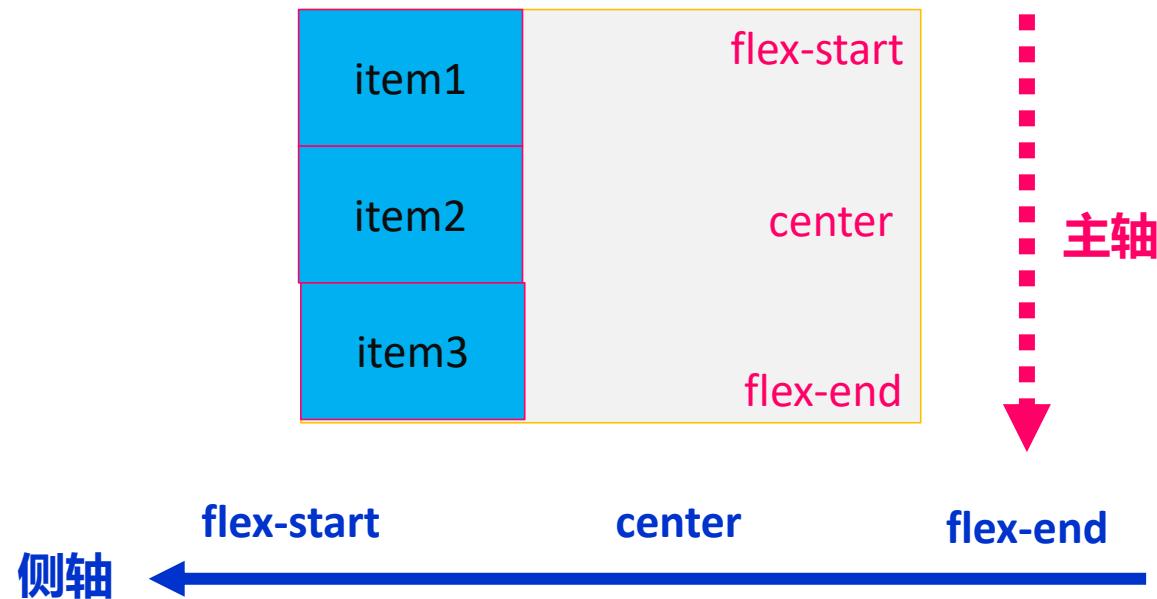
- 主轴默认方向，
 - flex-direction: row



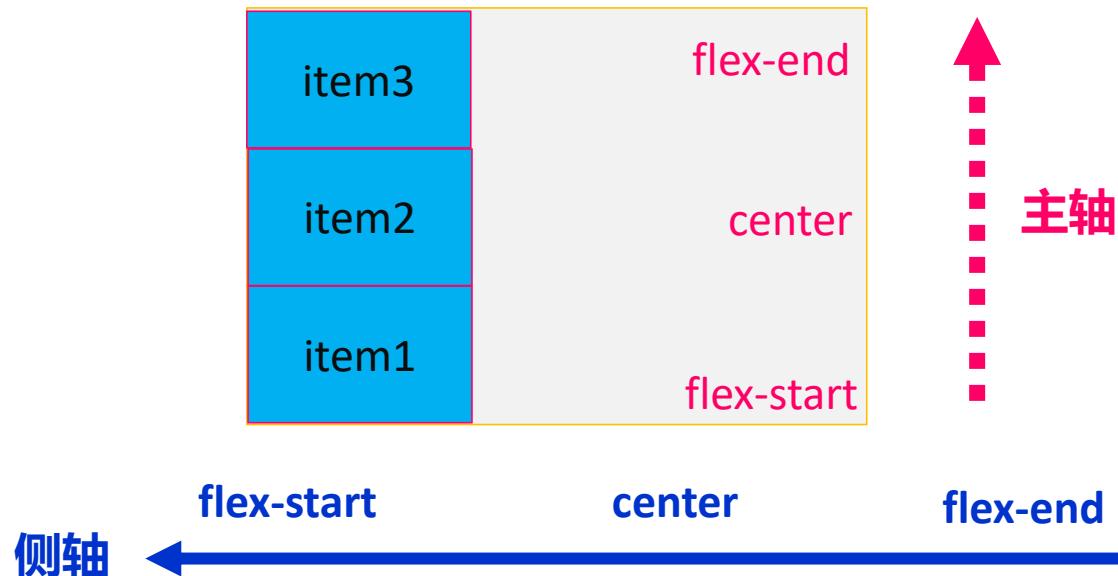
- 也可以设置
 - flex-direction: row-reverse



- 也可以通过设置
 - flex-direction: column

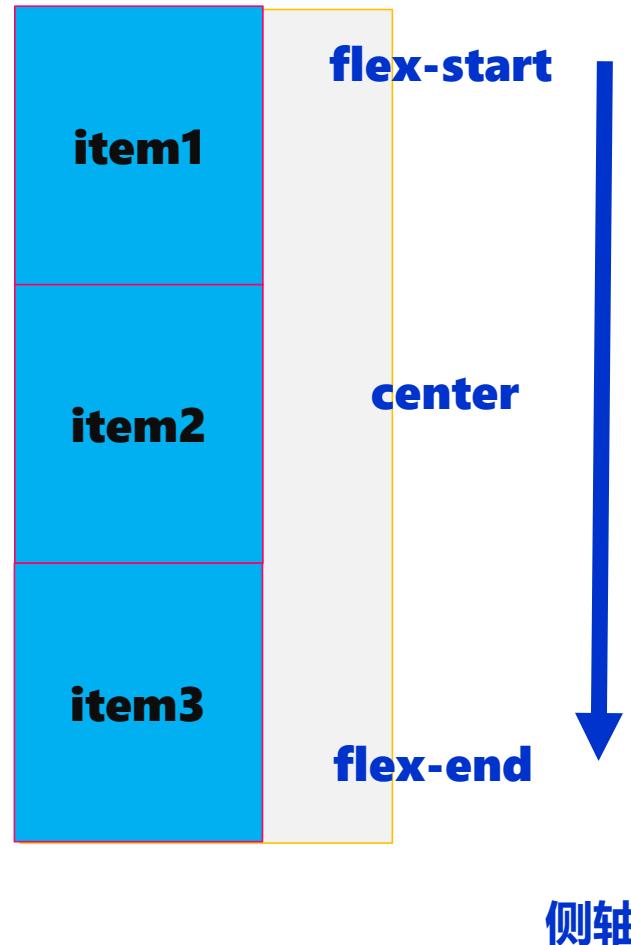


- 也可以设置
 - flex-direction: column -reverse

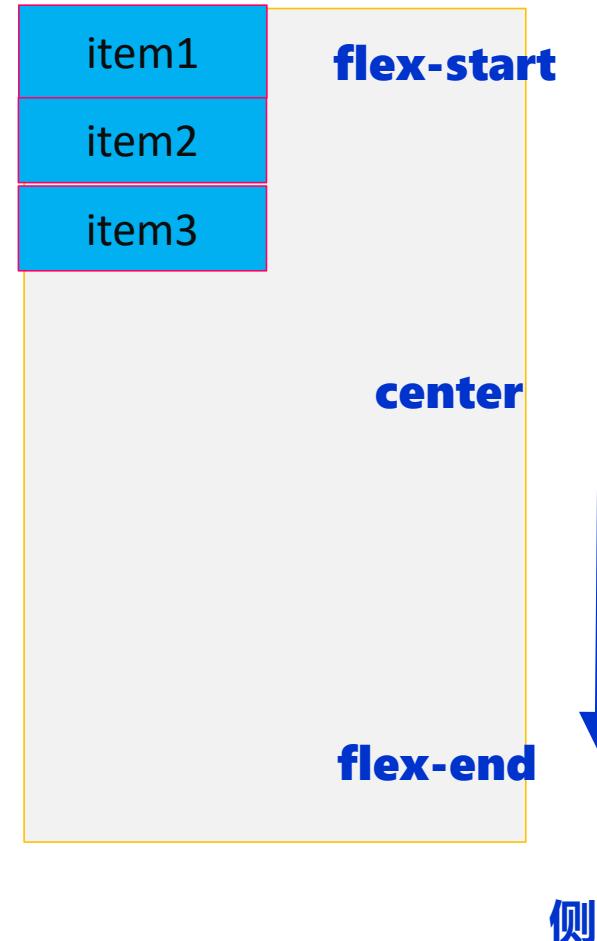


- 可以设置 flex-flow 来实现 flex-direction 和 flex-wrap 简写。
 - flex-flow : column wrap

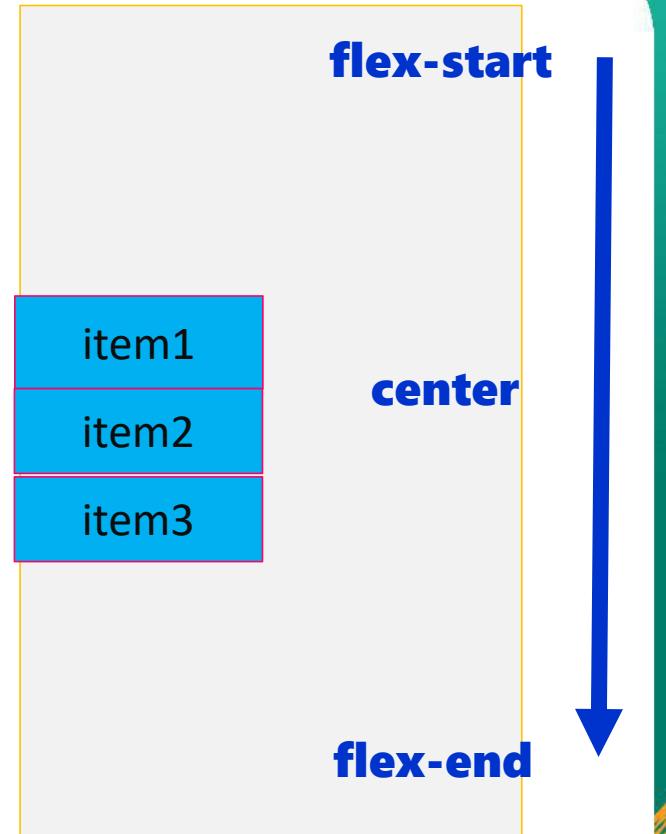
- 当伸缩容器在**主轴上没有足够的空间**容纳所有的伸缩项目，伸缩容器设置了flex-wrap: wrap，并且伸缩项目没有设置大小，完全由内容“撑起”，如同在伸缩项目中设置了
 - align-content: stretch
- 所有伸缩项目默认均匀分配侧轴空间



- 当伸缩容器在**主轴上没有足够的空间**容纳所有的伸缩项目，伸缩容器设置了flex-wrap: wrap，并且伸缩项目没有设置大小，完全由内容“撑起”，如果设置
 - align-content: flex-start

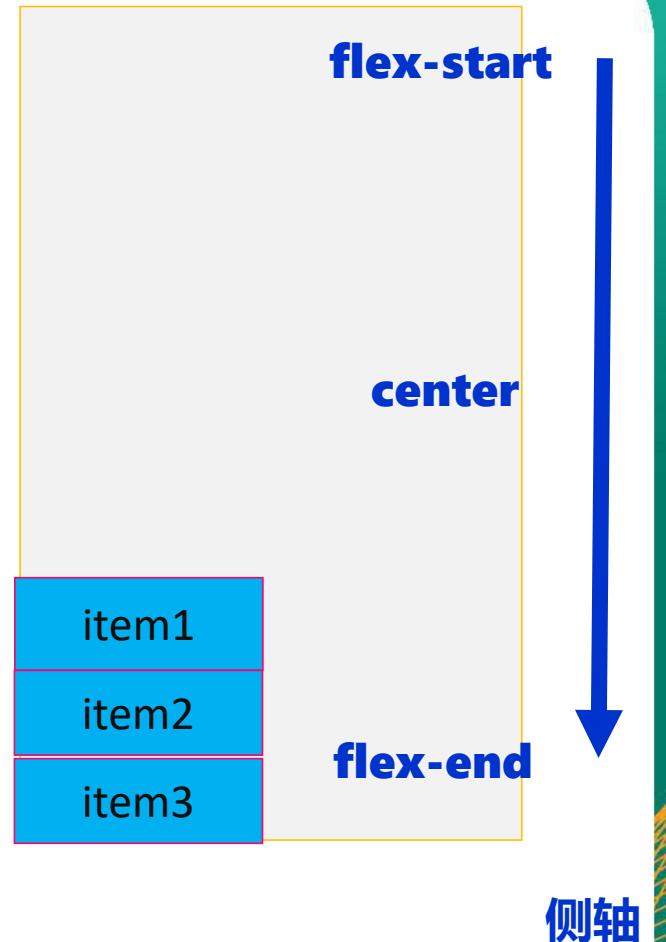


- 当伸缩容器在**主轴上没有足够的空间**容纳所有的伸缩项目，伸缩容器设置了flex-wrap: wrap，并且伸缩项目没有设置大小，完全由内容“撑起”，如果设置
 - align-content: center



侧轴

- 当伸缩容器在**主轴上没有足够的空间**容纳所有的伸缩项目，伸缩容器设置了flex-wrap: wrap，并且伸缩项目没有设置大小，完全由内容“撑起”，如果设置
 - align-content: flex-end
-

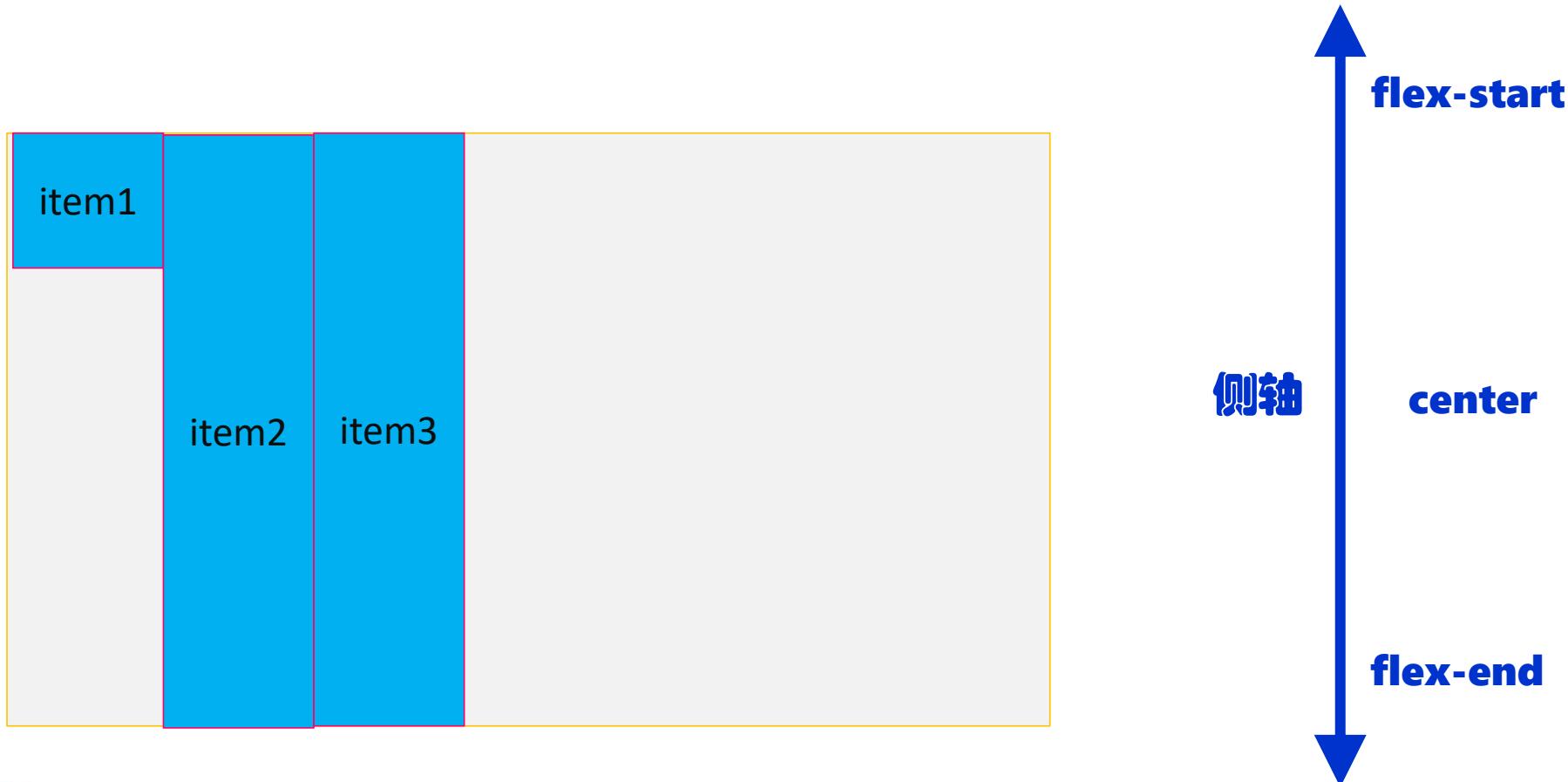


伸缩项目在侧轴排列

- 当伸缩容器足够大时，可以分别设置每个伸缩项目在伸缩容器侧轴的位置，例如默认值
 - align-self: stretch;



- 当伸缩容器足够大时，可以分别设置每个伸缩项目在伸缩容器侧轴的位置，例如为item1设置了
 - align-self: flex-start;



- 当伸缩容器足够大时，可以分别设置每个伸缩项目在伸缩容器侧轴的位置，例如为item1设置了
 - align-self: center;

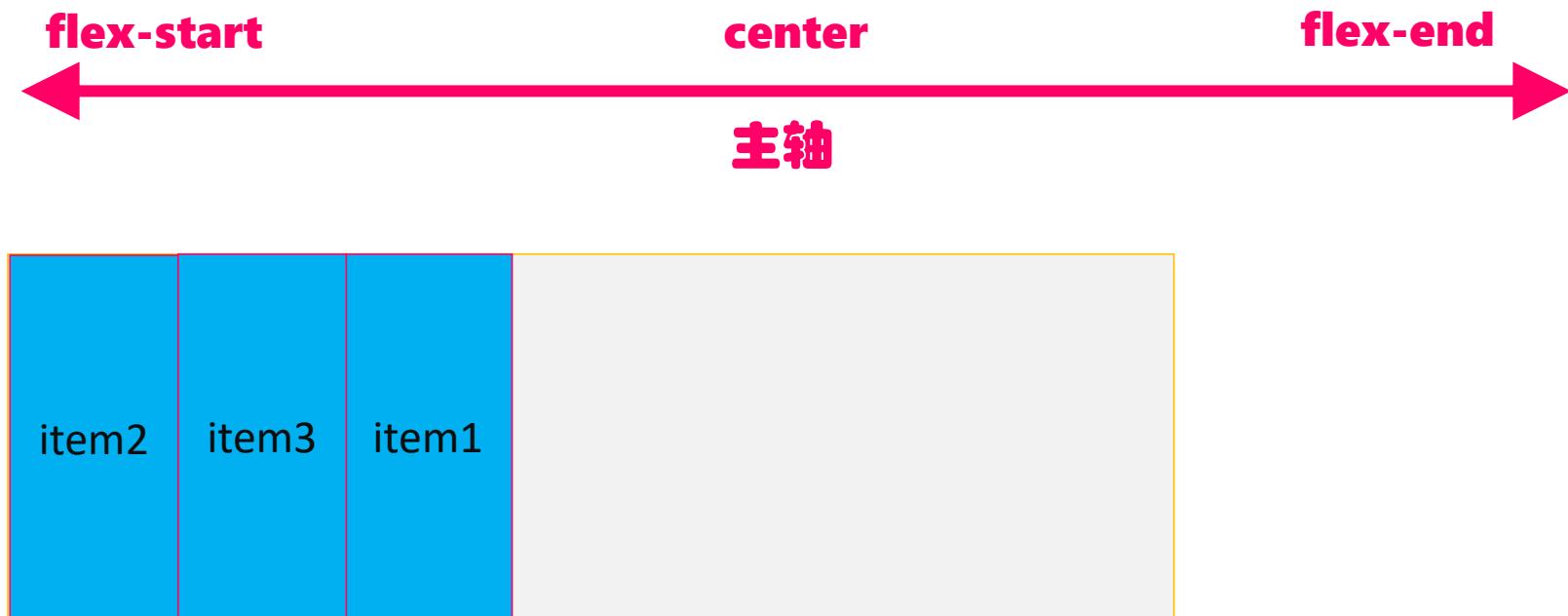


- 当伸缩容器足够大时，可以分别设置每个伸缩项目在伸缩容器侧轴的位置，例如为item1设置了
 - align-self: flex-end;

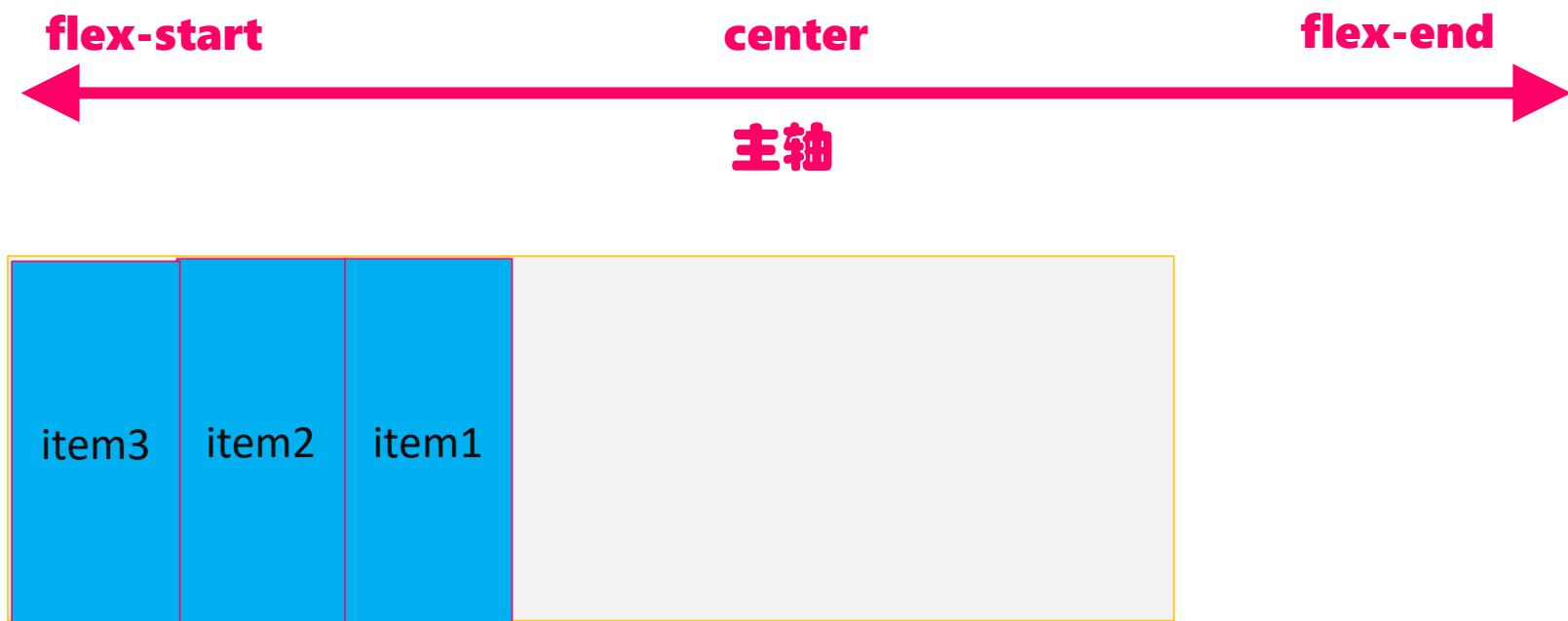


伸缩项目在主轴排列

- 当伸缩容器足够大时，可以分别设置每个伸缩项目在伸缩容器主轴的位置，例如为item1设置了
 - order: 1;

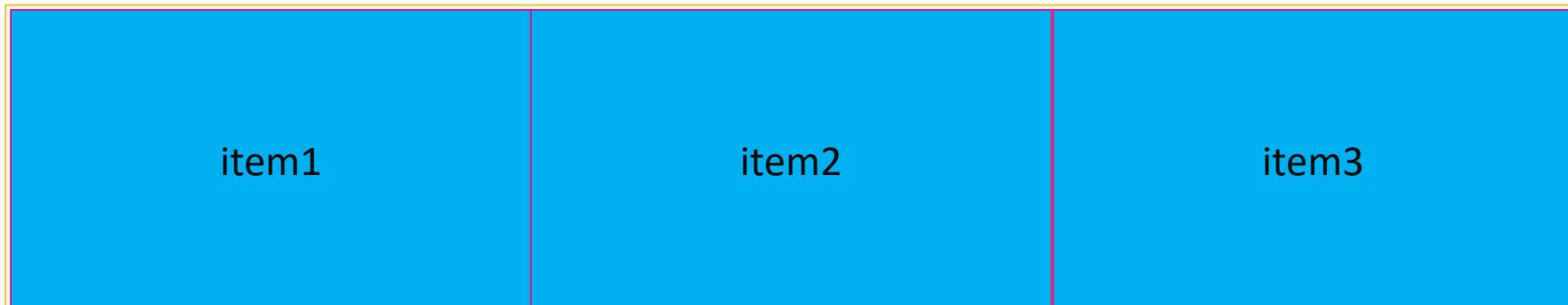


- 当伸缩容器足够大时，可以分别设置每个伸缩项目在伸缩容器主轴的位置，例如为item1, item2分别设置
 - order: 2;
 - order: 1;



伸缩项目 分配主轴剩余空间

- 当伸缩容器主轴空间足够大时，可以分别设置每个伸缩项目如何分配主轴空间。其属性为，
 - flex-grow
-



伸缩容器

width: 900px

伸缩项目1

flex-grow: 1

伸缩项目2

flex-grow: 1

伸缩项目3

flex-grow: 1

计算公式:

$$900 / (1 + 1 + 1) = 300$$

伸缩项目1
实际宽度
300px

伸缩项目2
实际宽度
300px

伸缩项目3
实际宽度
300px

伸缩容器

width: 900px

伸缩项目1

flex-grow: 2

伸缩项目2

flex-grow: 1

伸缩项目3

flex-grow: 1

计算公式:

$$900 / (2 + 1 + 1) = 225$$

伸缩项目1
实际宽度
510px

伸缩项目2
实际宽度
225px

伸缩项目3
实际宽度
225px

设置伸缩项目 初始宽度

- 有两种方式可以设置伸缩项目的初始宽度
 - ① width : length
 - ② flex-basis : length
- 使用flex-basis为元素指定宽度，效果与width相同
- 当元素设置了初始宽度，且伸缩容器主轴空间不足，伸缩项目会被压缩。

伸缩项目 如何被压缩

- 当设置伸缩项目大小，且伸缩容器无法在主轴方向上容纳所有的伸缩项目时，**每个伸缩项目都会被按比例被压缩**。默认如同设置了
 - flex-shrink: 1
- 如果不期望伸缩项目被压缩，可以设置
 - flex-shrink: 0
 - **以上设置会导致伸缩项目在主轴方向“溢出”**

伸缩容器

width: 400px

伸缩项目1

flex-shrink : 1

flex-basis: 100px



伸缩项目2

flex-shrink 2

flex-basis: 200px

伸缩项目3

flex-shrink :1

flex-basis: 300px

一、需要压缩的宽度

$$100 + 200 + 300 - 400 = 200$$

二、“总压缩基数”

$$100*1 + 200*2 + 300*1 = 800$$

三、每个伸缩项目实际压缩率

$$100*1 / 800 = 0.125$$

$$200*2 / 800 = 0.5$$

$$300*1 / 800 = 0.375$$

四、每个伸缩项目实际需要压缩空间

$$200 * 0.125 = 25$$

~~$$200 * 0.5 = 100$$~~

~~$$200 * 0.375 = 75$$~~

五、每个伸缩项目实际的宽度

$$100 - 25 = 75$$

$$200 - 100 = 100$$

$$200 - 75 = 225$$

伸缩项目1
实际宽度
75px

伸缩项目2
实际宽度
100px

伸缩项目3
实际宽度
225px





线性渐变&径向渐变

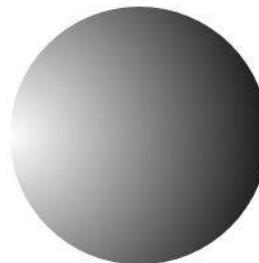
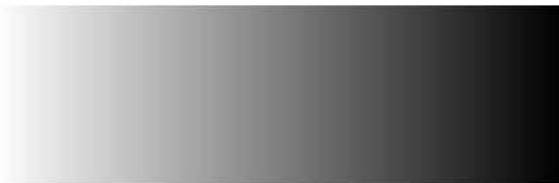
讲师：许井龙

ngsteel@qq.com

2016年第一版

- 一、线性渐变
 - 1. 颜色渐变方向&多颜色渐变
 - 2. 设置渐变色起始位置
 - 3. 线性渐变重复
- 二、径向渐变
 - 1. 基本用法
 - 2. 设置渐变形状-圆形 (圆心, 半径)
 - 3. 设置渐变形状-椭圆形 (圆心, 半径)
 - 4. 通过关键词隱式为径向渐变设置大小
 - 5. 径向渐变重复

- 线性渐变：颜色沿着一条直线轴变换。



- 径向渐变：从指定的圆心向外渐变。



线性渐变

- **linear-gradient()** 函数创建了一个呈现线性渐变的颜色的 `<image>`，在图像内无尺寸概念。也就是说其没大小概念。
- **radial-gradient()** 函数创建了一个呈现线性渐变的颜色的 `<image>`，在图像内无尺寸概念。也就是说其没大小概念。

```
linear-gradient(  
    [ <angle> | to <side-or-corner> , ]? <color-stop> [,  
<color-stop>]+ )
```

渐变的方向

渐变的颜色

- linear-gradient(to right, blue, red);
 - 颜色渐变方向：从左->右
- linear-gradient(to left top, blue, red);
 - 颜色渐变方向：从右下角 -> 左上角
- linear-gradient(45deg, blue, red);
 - 顺时针旋转45度。 (注意：默认是180deg)

距离：

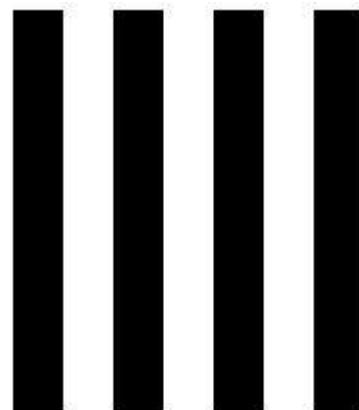
```
background: linear-gradient(  
    to right,  
    blue 10px,  
    red 50px  
)
```

设置每个颜色过渡的起始位置

- repeating-linear-gradient

示例代码：

```
background: repeating-linear-gradient(  
    to right,  
    transparent,  
    transparent 25px,  
    black 25px,  
    black 50px  
) ;
```



- 制作加载条



径向渐变

- 圆形或椭圆形渐变。颜色不再沿着一条直线轴变换，而是从一个“圆心”向外扩散。
- 径向渐变只有两种形状：椭圆形，圆形

基本用法（无需设置形状，默认椭圆形）：

```
radial-gradient(
```

```
    red, blue
```

```
);
```

设置渐变形状（圆形）

设置为圆形

```
radial-gradient(  
    circle,  
    red,  
    blue  
) ;
```

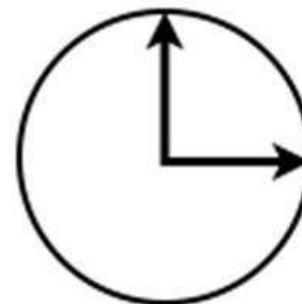
设置圆形圆心

```
radial-gradient(  
    circle at 100px 100px ,  
    red,  
    blue  
) ;
```

自定义圆形渐变范围的X轴， Y轴半径

设置圆形的半径

```
background: radial-gradient(  
    circle 200px at 100px 50px,  
    #fff,  
    #000  
);
```



设置为椭圆形

```
background: radial-
gradient(
    ellipse,
    blue 20px,
    red 30px ,
    yellow 50px
);
```

设置椭圆形的圆心

```
background: radial-
gradient(
    ellipse at 0px 0px,
    blue 20px,
    red 30px ,
    yellow 50px
);
```

自定义椭圆形渐变范围的X轴，Y轴半径

设置椭圆形X，Y轴半径

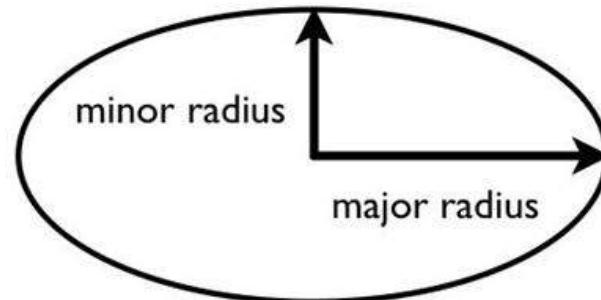
```
background: radial-gradient(
```

```
    ellipse 50px 100px at 100px 50px,
```

```
    #fff,
```

```
    #000
```

```
);
```



```
background: repeating-radial-gradient(  
    circle,  
    transparent,  
    transparent 10px,  
    black 10px,  
    black 20px  
);
```

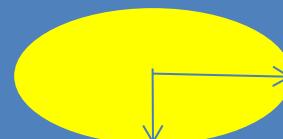


- 圆心规则：
 - closest-side: 渐变的圆心距离元素最近边的距离作为径向渐变半径。
 - closest-corner: 渐变的圆心距离元素最近角的距离作为径向渐变半径。
 - farthest-side: 渐变的圆心距离元素最近边的距离作为径向渐变半径。
 - farthest-corner: 渐变的圆心距离元素最近角的距离作为径向渐变半径。
- 椭圆形规则：
 - closest-side: 渐变的圆心距离元素最近两个边的距离作为径向渐变半径。
 - closest-corner: 渐变的圆心距离元素最近角两个边的距离作为径向渐变半径。
 - farthest-side: 渐变的圆心距离元素最近两个边的距离作为径向渐变半径。
 - farthest-corner: 渐变的圆心距离元素最近角的两个边距离作为径向渐变半径。

closest-side



closest-side



示例代码：

```
background: repeating-radial-gradient(  
    ellipse farthest-corner,  
    red,  
    black 5%,  
    blue 5%,  
    green 10%  
) ;
```



具有过渡效果的CSS属性

background-color	font-size
background-position	font-weight
border-bottom-color	height
border-bottom-width	left
border-left-color	letter-spacing
border-left-width	line-height
border-right-color	margin-bottom
border-right-width	margin-left
border-spacing	margin-right
border-top-color	margin-top
border-top-width	max-height
bottom	max-width
clip	min-height
color	min-width

opacity

outline-color

outline-width

padding-bottom

padding-left

padding-right

padding-top

right

text-indent

text-shadow

top

vertical-align

visibility

width

word-spacing

z-index

表达式	首次定义在	含义
*	2	通配符选择器，选择所有页面元素。
E	1	元素选择器
E[foo]	2	元素属性选择器，元素包含属性foo
E[foo="bar"]	2	元素属性选择器，元素包含属性foo，属性foo=bar
E[foo~="bar"]	2	选择foo属性包含单词 "bar" 的元素（bar的前后必须有空格），并设置其样式；
E[foo^="bar"]	3	元素E属性foo以bar开头
E[foo\$="bar"]	3	元素E属性foo以bar结尾
E[foo*="bar"]	3	元素E属性foo包含bar
E[foo ="en"]	2	选择器用于选取带有以指定值开头的属性值的元素。 该值必须是整个单词，比如 lang="en"（此情况，lang只能有一个en属性值），或者后面跟着连字符，比如 lang="en-us"。
E:root	3	根元素选择器
E:nth-child(n)	3	元素E的位于兄弟元素的第n位
E:nth-last-child(n)	3	元素E的位于兄弟元素的倒数第n位
E:nth-of-type(n)	3	元素E的位于相同类型兄弟元素的第n位
E:nth-last-of-type(n)	3	元素E的位于相同类型兄弟元素的倒数第n位
E:first-child	2	元素E的第一个子元素
E:last-child	3	元素E的位于兄弟元素的最后位置
E:first-of-type	3	元素E的位于相同类型兄弟元素的最前位置
E:last-of-type	3	元素E的位于相同类型兄弟元素的最后位置
E:only-child	3	元素E是其父元素唯一子元素
E:only-of-type	3	元素E是其父元素唯一指定类型子元素（可以有其他类型兄弟元素）
E:empty	3	元素E没有子元素，当然也不能包含文本。
E:link	1	未被访问的链接
E:visited	1	已经访问的链接
E:active	1 and 2	元素被点击
E:hover	1 and 2	鼠标悬浮在元素上
E:focus	1 and 2	元素获取焦点
E:invalid,	3	INPUT元素验证不通过显示的样式
E:valid	3	INPUT元素验证通过显示的样式
E:required	3	INPUT元素必填时提示
E:target	3	E元素作为其他元素的目标元素时
E:lang(fr)	2	E元素lang属性的值是fr
E:enabled	3	元素E处于enabled或者disabled时
E:disabled	3	元素E处于enabled或者disabled时
E:checked	3	元素E处于checked状态时
E::first-line	1	元素的第一行
E::first-letter	1	元素的第一个字母
E::before	2	在元素E前创建一个伪元素
E::after	2	在元素E后创建一个伪元素
E.warning	1	E元素的class列表中包含有.warning
E#myid	1	E元素ID为myid
E:not(s)	3	E元素没有使用s选择器的所有元素
E F	1	后代选择器
E > F	2	子元素选择器
E + F	2	紧邻兄弟选择器
E ~ F	3	一般兄弟选择器

备注：CSS选择器

类别
全局选择器
类型选择器
属性选择器
连接伪类
用户动作伪类
状态伪类
状态伪类
状态伪类
目标伪类选择器
语言伪类
语言代码
状态伪类
状态伪类
状态伪类
伪元素
伪元素
伪元素
伪元素
类选择器
ID选择器
否定伪类
后代选择器
子元素选择器
紧邻兄弟元素选择器
一般兄弟元素选择器



CSS3 字体及自定义字体

讲师：许井龙

ngsteel@qq.com

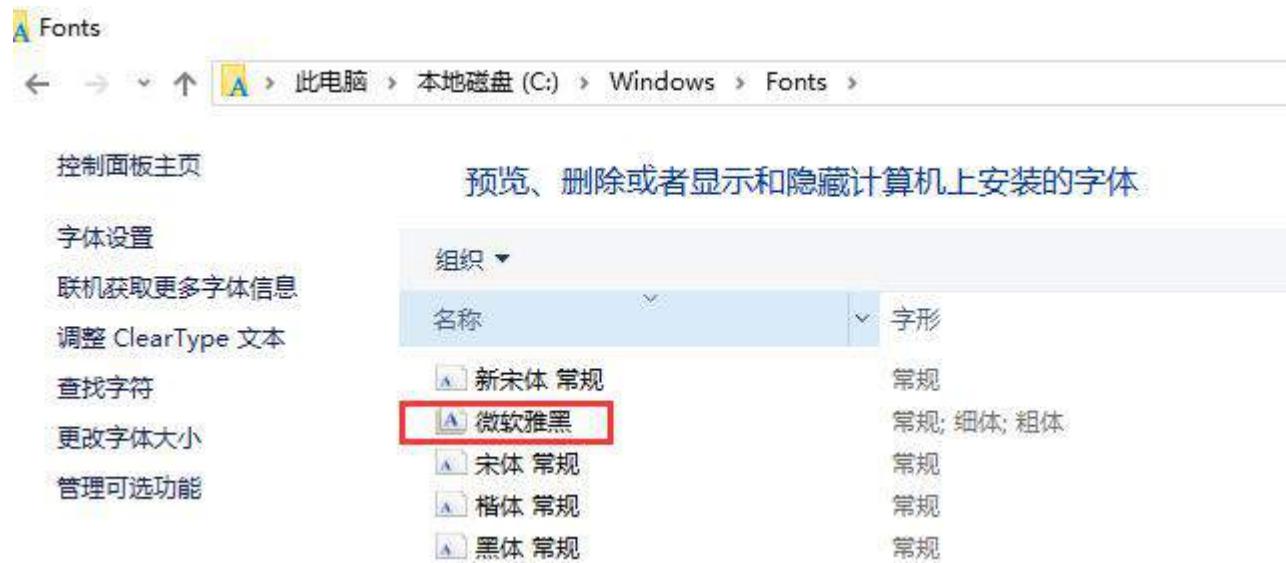
2016年第一版

- 字体 和 文本 区别
- 默认字体
- 免费英文字体
- 免费中文字体

- 字体 (font) : 字的形体结构，由自身的展示样式。
- 文本 (text) : 就是一组字或者字符串。

font-family来源

1、用户机器安装的字体（安全字体）



font-family来源

2、保存在自己服务器的字体

@font-face是CSS3中的一个模块，他主要是把自己定义的Web字体嵌入网页中。

```
@font-face {  
    font-family: <YourWebFontName>;  
    src: <source> [<format>][,<source> [<format>]]*;  
    [font-weight: <weight>];  
    [font-style: <style>];  
}
```

取值说明

1、YourWebFontName:此值指的就是你自定义的字体名称，最好是使用你下载的默认字体，他将被引用到你的Web元素中的font-family。如“font-family: "YourWebFontName";”

2、source:此值指的是你自定义的字体的存放路径，可以是相对路径也可以是绝路径；

3、format: 此值指的是你自定义的字体的格式，主要用来帮助浏览器识别，其值主要有以下几种类型：truetype,opentype,truetype-aat,embedded-opentype,avg等；

4、weight和style:这两个值大家一定很熟悉，weight定义字体是否为粗体，style主要定义字体样式，如斜体。

兼容浏览器

一、TrueType(.ttf)格式：

.ttf字体是Windows和Mac的最常见的字体，是一种RAW格式，因此他不为网站优化，支持这种字体的浏览器有【IE9+, Firefox3.5+, Chrome4+, Safari3+, Opera10+, iOS Mobile Safari4.2+】；

二、OpenType(.otf)格式：

.otf字体被认为是一种原始的字体格式，其内置在TrueType的基础上，所以也提供了更多的功能，支持这种字体的浏览器有【Firefox3.5+, Chrome4.0+, Safari3.1+, Opera10.0+, iOS Mobile Safari4.2+】；

三、Web Open Font Format(.woff)格式：

.woff字体是Web字体中最佳格式，他是一个开放的TrueType/OpenType的压缩版本，同时也支持元数据包的分离，支持这种字体的浏览器有【IE9+, Firefox3.5+, Chrome6+, Safari3.6+, Opera11.1+】；

四、Embedded Open Type(.eot)格式：

.eot字体是IE专用字体，可以从TrueType创建此格式字体，支持这种字体的浏览器有【IE4+】；

五、SVG(.svg)格式：

.svg字体是基于SVG字体渲染的一种格式，支持这种字体的浏览器有【Chrome4+, Safari3.1+, Opera10.0+, iOS Mobile Safari3.2+】。

下载免费字体

<http://www.dafont.com/>

Fancy	Fire, Ice	Foreign look	Techno	Gothic	Basic	Script	Dingbats	Bar Code	Holiday
Cartoon	Decorative	Chinese, Jpn	Square	Medieval	Sans serif	Calligraphy	Alien	Nature	Valentine
Comic	Typewriter	Arabic	LCD	Modern	Serif	School	Animals	Sport	Easter
Groovy	Stencil, Army	Mexican	Sci-fi	Celtic	Fixed width	Handwritten	Asian	Heads	Halloween
Old School	Retro	Roman, Greek	Various	Initials	Various	Brush	Ancient	Kids	Christmas
Curly	Initials	Russian		Various		Trash	Runes, Elvish	TV, Movie	Various
Western	Grid	Various				Graffiti	Esoteric	Logos	
Eroded						Old School	Fantastic	Sexy	
Distorted						Various	Horror	Army	
Destroy							Games	Mus	
Horror							Shapes	Vario	

Fancy > Fire, Ice

1 2 3 4 ►

Preview Fonts Size Sort by

 Show variants

Blazed  by Bright Ideas

2,736,535 downloads (432 yesterday) 41 comments

[Download](#)



下载解压缩后，我们看到如下文件。



.ttf字体是Windows和Mac的最常见的字体，是一种RAW格式，因此他不为网站优化。

支持这种字体的浏览器有【IE9+, Firefox3.5+, Chrome4+, Safari3+, Opera10+, iOS Mobile Safari4.2+】。

生成不同版本浏览器支持的字体文件

<https://www.fontsquirrel.com/tools/webfont-generator>



The screenshot shows the FontSquirrel website. At the top left, it says "100% Free For Commercial Use." and has the "FONT SQUIRREL" logo with a squirrel icon. Below the logo are four small thumbnail images: a bag of walnuts, a RAM module, a red box, and a bottle of water. A navigation bar at the bottom includes links: Hot, Recent, Almost Free, Shop, Generator (which is highlighted in blue), Font Identifier, Font Talk, and Blog.

Webfont Generator

Usage: Click the "Upload Fonts" button, check the agreement and download your fonts. If you need more fine-grain control, choose the **Expert** option.

UPLOAD FONTS ↑

You currently have no fonts uploaded.

BASIC

Straight conversion with minimal processing.

OPTIMAL

Recommended settings for performance and speed.

EXPERT...

You decide how best to optimize your fonts.

Webfont Generator

Usage: Click the "Upload Fonts" button, check the agreement and download your fonts. If you need more fine-grain control, choose the **Expert** option.

UPLOAD FONTS 

You currently have no fonts uploaded.

BASIC

Straight conversion with minimal processing.

OPTIMAL

Recommended settings for performance and speed.

EXPERT...

You decide how best to optimize your fonts.

Agreement:

Yes, the fonts I'm uploading are legally eligible for web embedding.

Font Squirrel offers this service in good faith. Please honor the EULAs of your fonts.

Do you find this service useful? Support us with a

FONT SQUIRREL SHIRT →

Webfont Generator

Usage: Click the "Upload Fonts" button, check the agreement and download your fonts. If you need more fine-grain control, choose the **Expert** option.

UPLOAD FONTS

Blazed Regular

ttf

246 glyphs

75 KB



BASIC

Straight conversion with minimal processing.

OPTIMAL

Recommended settings for performance and speed.

EXPERT...

You decide how best to optimize your fonts.

Agreement:

Yes, the fonts I'm uploading are legally eligible for web embedding.

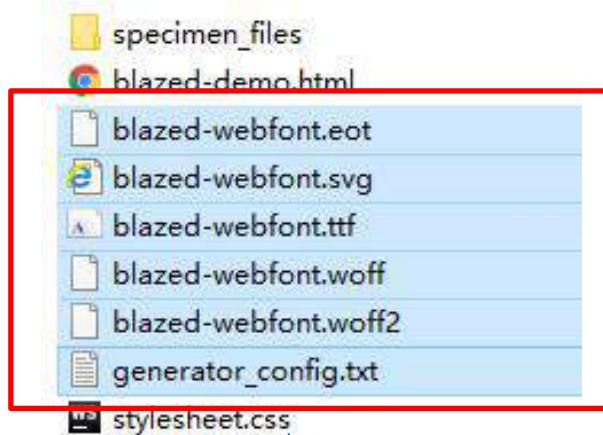
Font Squirrel offers this service in good faith. Please honor the EULAs of your fonts.

DOWNLOAD YOUR KIT

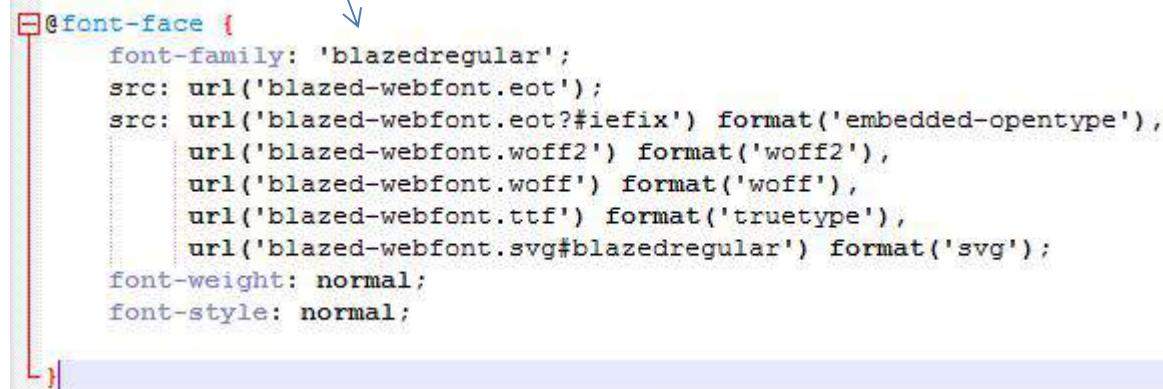
Do you find this service useful? Support us with a

FONT SQUIRREL SHIRT →

下载后的kit解压缩后，

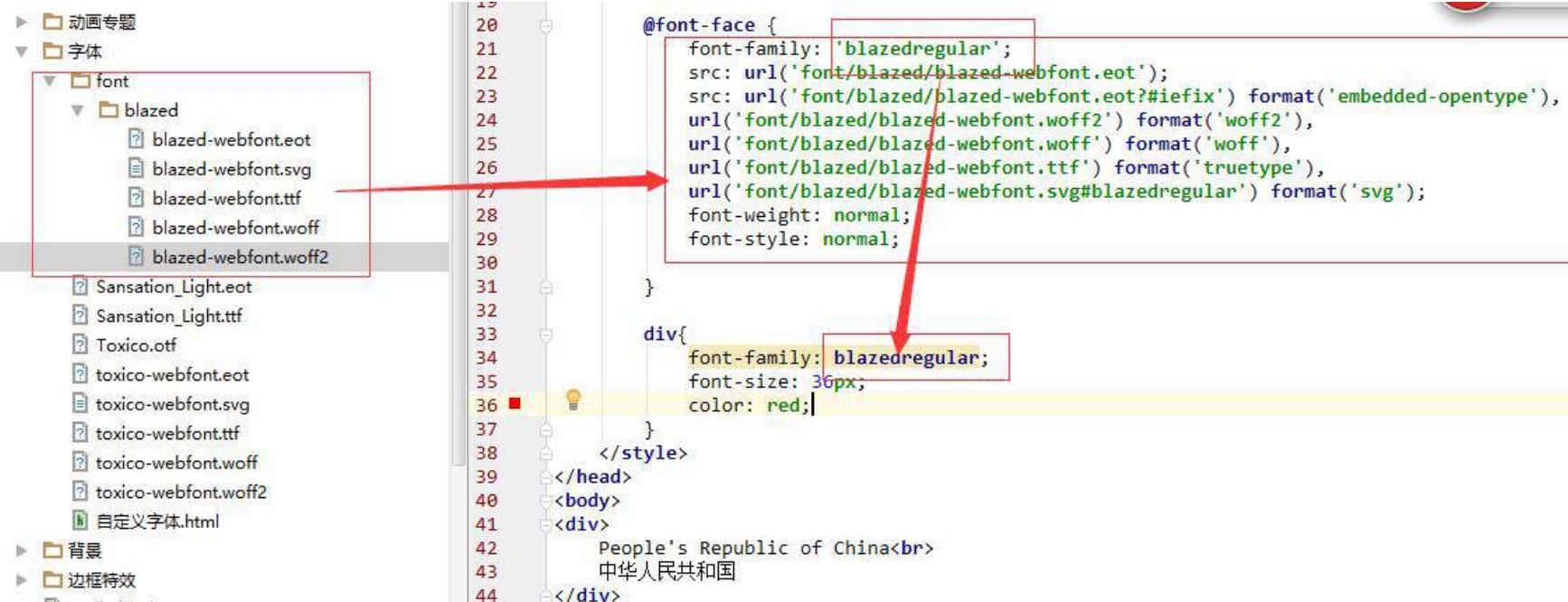


所有的字体



```
@font-face {
    font-family: 'blazedregular';
    src: url('blazed-webfont.eot');
    src: url('blazed-webfont.eot?#iefix') format('embedded-opentype'),
         url('blazed-webfont.woff2') format('woff2'),
         url('blazed-webfont.woff') format('woff'),
         url('blazed-webfont.ttf') format('truetype'),
         url('blazed-webfont.svg#blazedregular') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

项目中使用



The screenshot shows a code editor with a file structure on the left and code snippets on the right.

File Structure:

- 动画专题
- 字体
 - font
 - blazed
 - blazed-webfont.eot
 - blazed-webfont.svg
 - blazed-webfont.ttf
 - blazed-webfont.woff
 - blazed-webfont.woff2
 - Sensation_Light.eot
 - Sensation_Light.ttf
 - Toxico.otf
 - toxico-webfont.eot
 - toxico-webfont.svg
 - toxico-webfont.ttf
 - toxico-webfont.woff
 - toxico-webfont.woff2
 - 自定义字体.html

Code Snippets:

```
17
20 @font-face {
21   font-family: 'blazeregular';
22   src: url('font/blazed/blazed-webfont.eot');
23   src: url('font/blazed/blazed-webfont.eot?#iefix') format('embedded-opentype'),
24        url('font/blazed/blazed-webfont.woff2') format('woff2'),
25        url('font/blazed/blazed-webfont.woff') format('woff'),
26        url('font/blazed/blazed-webfont.ttf') format('truetype'),
27        url('font/blazed/blazed-webfont.svg#blazeregular') format('svg');
28   font-weight: normal;
29   font-style: normal;
30 }
31
32
33
34 div{
35   font-family: blazeregular;
36   font-size: 36px;
37   color: red;
38 }
39
40
41
42 People's Republic of China<br>
43 中华人民共和国
44 
```

A red box highlights the font-face rule, and a red arrow points from it to the div selector in the CSS. Another red box highlights the 'blazeregular' font-family declaration within the div selector, and a red arrow points from it to the corresponding font-family declaration in the font-face rule.

效果，所有浏览器
都支持

People's Republic of China
中华人民共和国

关于中文字体，以Window10为例，

C:\Windows\Fonts

此电脑 > 本地磁盘 (C:) > Windows > Fonts

预览、删除或者显示和隐藏计算机上安装的字体

组织 预览 删除 隐藏

名称	字形	显示/隐藏	设计用于	类别	设计者/制造商
新宋体 常规	常规	显示	中文(简体中文)	文本	Beijing ZhongYi Electronics, Co.
微软雅黑	常规; 细体; 粗体	显示	中文(简体中文)	文本	Microsoft Corporation
宋体 常规	常规	显示	中文(简体中文)	文本	Beijing ZhongYi Electronics, Co.
楷体 常规	常规	显示	中文(简体中文)	文本	Beijing ZhongYi Electronics Co.
黑体 常规	常规	显示	中文(简体中文)	文本	Beijing ZhongYi Electronics, Co.
仿宋 常规	常规	显示	中文(简体中文)	文本	Beijing ZhongYi Electronics Co.
等线	常规; 细体; 粗体	显示	中文(简体中文)	文本	Founder Corporation
Yu Gothic UI	常规; 半粗体; 细体; 半细体; 粗体	隐藏	日文	文本	Jiyukobo Ltd.
Yu Gothic	常规; 细体; 中等; 粗体	隐藏	日文	文本	Jiyukobo Ltd.
Wingdings 常规	常规	显示	符号	符号/象形文字	Microsoft Corporation
Webdings 常规	常规	显示	符号	符号/象形文字	Microsoft Corporation
Verdana	常规; 粗体; 粗斜体; 斜体	显示	拉丁文; 希腊语; ...	文本	Carter + Cone
Trebuchet MS	常规; 粗体; 粗斜体; 斜体	显示	拉丁文; 希腊语; ...	显示	Microsoft Corporation
Times New Roman	常规; 粗体; 粗斜体; 斜体	显示	拉丁文; 希腊语; ...	文本	The Monotype Corporation
Terminal 常规	常规	显示	拉丁文	文本	Microsoft Corporation
Tahoma	常规; 粗体	显示	拉丁文; 希腊语; ...	文本	Microsoft Corporation

浏览器都可以使用。

font-family来源

3、第三方网站字体



The screenshot shows the homepage of the YouZiku online font library. At the top, there's a navigation bar with links like '应用', '设置', '素材网站', etc. Below it is a search bar with a placeholder '关键词搜索' and a blue '搜索' button. The main header features a stylized pear icon and the text '有字库 WEB FONT SERVER'. A secondary navigation bar below the main one has tabs for '首页', '在线字库' (which is currently selected), '定制字库', '帮助中心', and '最新动态'. In the center, there's a large search input field. Below the search area, there are four categories: '所有字库', '正版授权', '常用推荐', and '私有字体'. Under '正版授权', there are filters for '类别' (Type) and '子类别' (Sub-category). The '语系' (Script) filter under '类别' and the '中文' (Chinese) filter under '子类别' are both highlighted with red circles. At the bottom, there are sorting options: '使用最多', '推荐最多', '最新上传', '评论最多', and '收藏最多'.



德彪钢笔行书字库 龙飞凤舞 中国文字...

使用

收藏

485 收藏 / 5 推荐 / 41775 使用 / 20 评论

[德彪钢笔行书字库](#) | 行书 | 硬笔手写

» 调用方法

CSS模式

卢教模式

这里有个坑，你的网页用几个中文字，你这里就写几个。

第一步：生成字体 (输入需要使用的中文字)

中华人民共和国

生成

当前为CSS引用方式

1. 在要使用该字体的标签属性中添加如下代码：

```
font-family: 'LiDeBiao-Xing32ed958a385699';
```

或调用class

```
class="css2ed958a385699"
```

例如：<h1 style="font-family:'LiDeBiao-Xing32ed958a385699';">你的内容</h1>

例如：<div class="css2ed958a385699"> 你的内容</div>

2. 引用方式（分为同步和异步两种方式，根据需要选择其中一种方式）

同步方式（适用于少量文本）

将以下代码加到你网页的<head>标签之内。

```
<link href='http://api.youziku.com/webfont/CSS/571cc7bff629d80b1866fbfe' rel='stylesheet' type='text/css' />
```

异步加载方式（适用于大量文本）

将以下代码添加到您网页的</body>标签之后，</html>标签之前。

```
<script type="text/javascript" src="http://api.youziku.com/wwwroot/js/g_webfont.min.js"></script>

<script type="text/javascript">
    WebFont.load({
        custom: {
            urls: [ 'http://api.youziku.com/webfont/CSS/571cc7bfff629d80b1866fbfe' ]
        }
    });
</script>
```

提醒：

- 若是文字较少，推荐使用同步的方式。
- 异步引用方式会先显示整个页面，字体部分以默认字体显示（IE）或者空白（chrome），然后再加载字体效果，适用于页面中引用了大量文字的情况，以加快页面显示的速度，优化用户体验。
- 若是有多套字体，则直接在urls变量中添加url就可以了，比如有三套字体，则为urls:['url1','url2','url3']





CSS3 文本

讲师：许井龙

ngsteel@qq.com

2016年第一版

- 一. 文本换行
- 二. 文本阴影
- 三. 文本溢出

一、文本换行

- **overflow-wrap** : 浏览器为了防止文本溢出，是否可以在单词内换行。
 - **normal** 默认情况，浏览器只在 “空格” 或者 “半角”的位置换行。
 - **break-word** 为防止长文本（[例如网页地址](#)、长的单词）溢出，浏览器自行决定在何处 “截断” 单词。默认情况是禁止浏览器截断单词。

一、文本换行

- **word-break:** 让文本在任何位置换行
 - normal 根据语言自己的规则确定换行方式，中文到边界上的汉字换行，英文整个单词换行。
 - **break-all** 为了保证每一行的**空间不浪费**，强行“截断”英文单词。
 - **keep-all** 不准许“截断”。如果是中文把前后标点符号（全角）后开始换行，英文单词整个换行。
- IE6及以上浏览器对该属性支持。

一、文本换行

- **white-space**: 处理元素中的空白符 (CSS3新增)
 - **normal** 默认值。空白处会被浏览器忽略 (即多余空格被浏览器删除)。只保留正常的空格。
 - **pre**: 文本空白会被浏览器保留。其行为方式类似HTML的<pre>元素效果。
 - **pre-line**: 与normal类似，空白处会被浏览器忽略。不同点是保留换行符，IE7及以下浏览器不支持此属性。
 - **pre-wrap**: 保留空白符序列，换行单独一行显示，IE7及以下浏览器不支持此属性。
 - **nowrap**: 文本不换行，文本在同一行显示，直到遇到
位置。空白处会被浏览器忽略

二、文本阴影

- text-shadow: offset-x | offset-y | blur-radius | color
 - offset-x X轴偏移量
 - offset-y Y轴偏移量
 - blur-radius 模糊半径
 - color 阴影颜色
- IE8及以下浏览器兼容方案。
- filter: shadow([Color], [Direction], [Strength]);
 - Color 阴影颜色
 - Direction 设定投影的方向。0 阴影在文本上；45度阴影在文本右上角。
 - Strength



小练习：烟火效果的字体

三、文本溢出

- **text-overflow: ellipsis;** 当文本溢出时，省略号显示。
- **text-overflow: clip;** 仅仅简单的剪裁，不使用省略号。
- 注意：上述两个属性需要配合
 - white-space: nowrap; 和
 - overflow: hidden 一起使用，且容器需要定义宽度。
 - 只在块元素内生效。
- IE浏览器对齐支持非常好。



小练习：制作侧边栏列表





CSS3 元素选择器

讲师：许井龙

ngsteel@qq.com



- 一. :target 目标伪类选择器
 - 1. 巩固练习：文章目录
 - 2. 巩固练习：图片切换
- 二. 结构化伪类选择器
 - 1. 巩固练习：尺子
- 三. :checked伪类选择器
 - 1. 扩展知识：appearance 样式
- 四. 属性选择器
- 五. 伪元素选择器
 - 1. ::before
 - 2. ::after
 - 3. ::first-letter
 - 4. ::first-line
 - 5. ::selection
 - 6. 扩展知识：counter(), counters()
- 六. 多媒体选择器
 - 1. @media
 - 2. 扩展知识：attr()

七、兄弟元素选择器

当用户点击a元素，该元素锚链接指向的元素会显示在视口范围

```
<a href= "#e1" >一个锚链接 </a>
```

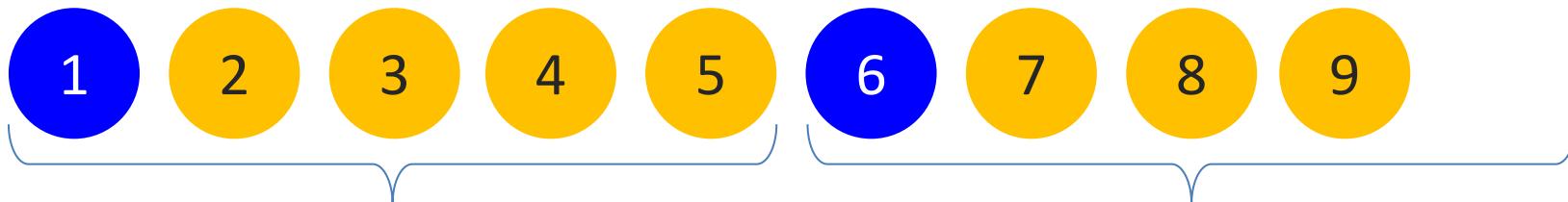
```
<p id= "e1" >  
当用户点击a元素，如果元素的href= "#e1"，此时我具有:target状态  
</p>
```

```
p:target{  
    //编写你的CSS样式  
}
```

- 文章列表
- 图片切换

- 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，所有被该选择器选择的层级)**，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

$5n+1$



分组方向

$5n+4$



分组方向

$-5n+8$ 

分组方向

编号-1, 0的两个元素不是真正的
元素, 占位用

 $5n-1$ 

分组方向

- E:nth-child(参数),
 - E 表示CSS选择器。E选择器选中的元素，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. **a n + b** 从第b个元素开始，每a个一组，选取第一个元素

```
span:nth-of-type(even) {  
    color: red;  
}
```

```
<div class="box">
```



1111
2222

3333
4444

5555
66666
77777

1 2222

<p>3333</p>

2 4444

<p>5555</p>

<p>666</p>

3 7777

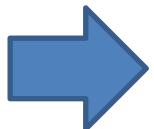
```
</div>
```

- 1.选中同一层级的所有span元素。
- 2.在选中后的span，重新计算位置，重新设置位置中位于偶数位置。

```
span:nth-child(even) {  
    color: red;  
}
```

```
<div class="box">
```

- 1 <div>1111</div>
 - 2 2222
 - 3 <p>3333</p>
 - 4 4444
 - 5 <p>5555</p>
 - 6 <p>666</p>
 - 7 7777
- ```
</div>
```



1111

2222

3333

4444

5555

77777

66666

选中同一层级的所有元素。

在同一层级中位置必须在偶数位。

	E:nth-of-type	E:nth-child
1	E选择器选中的所有元素，且被选中的元素都在同一DOM层级。	
2	E选择器选中的所有元素， <b>重新设置位置序号</b> 。(可以理解为，忽略其他不同类型元素的存在)。	E选择器选中的所有元素，被选中的元素必须在指定的位置且类型也必须相同

- : checked 当input type= 'checkbox' 被选中时。

```
input[type=checkbox] {
 -webkit-appearance: none;
 -moz-appearance: none;
 appearance: none;
}
```

**使用CSS3的appearance属性改变元素的外观**

- [attribute] 选择器用于选取带有指定属性的元素。

```
<input type="checkbox" name="city" id="city">
```

```
[type] {
 选取所有含有type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。

```
<input type="checkbox" name="city" id="city">
```

```
input[type=checkbox] {
 选取input元素，但其type属性值必须实checkbox
}
```

或者

```
input[name=city] {
 选取input元素，但其name属性值必须实city
}
```

- [attribute<sup>^</sup>=value] 选择器匹配属性值以指定值开头的每个元素。

```
<input type="checkbox" name="city" id="city" lang="en" >
```

```
input[type^=che] {
```

选取input元素，但其type属性值必须实che开头

```
}
```

- [attribute\$=value] 选择器匹配属性值以指定值结尾的每个元素。

```
<input type="checkbox" name="city" id="city" lang="en" >
```

```
input[type$=box] {
```

选取input元素，但其type属性值必须实box结尾

```
}
```

- [attribute $\sim$ =value] 选择器用于选取属性值中包含指定词汇的元素。

```
<div class="box box1">.box1</div>
```

```
div[class \sim =box] {
```

选取div元素，但其class属性值必须包含box单词（一个单词前后必须有空格）

```
}
```

- **[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"。

```
<div class="box box3" lang="en">.box3</div>
<div class="box box4" lang="en-us">.box4</div>
<div class="box box5" lang="us en">.box5</div>
```

```
div [lang|=en]{
 此时只有box3 和 box4 被选中
}
```

```
<div class=box1></div>
```

```
.box1::before{
 content: "Hello";
}
```

Hello

World

```
<div class=box1></div>
```

```
.box1::after{
 content: "Hello";
}
```

```
World
```

Hello

1. :: 是规范要求，实际开发中使用：，目的是得到IE8的支持
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.

1. 解决子元素浮动造成的父元素高度塌陷。
2. 页面中“重复出现”的文本内容，且不需要JS操作。
3. 客制化目录前的序号或者符号[ 配合CSS3 新增的*counter()*,  
*counters()*一起使用 ]

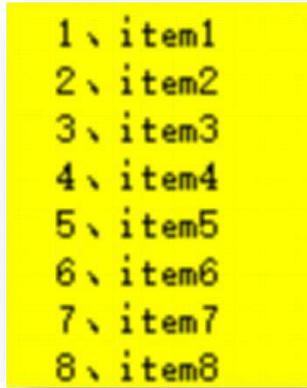
```

 item1
 item2
 item3
 item4

```



```
ul>li{
 counter-increment: num;
 list-style: none;
}
ul>li:before{
 content: counter(num) "、 ";
}
```



```
1、 item1
2、 item2
3、 item3
4、 item4
5、 item5
6、 item6
7、 item7
8、 item8
```

```

 item1
 item2

 item-sub1
 item-sub2
 item-sub3

 item3
 item4

```



```
ul{
 counter-reset: num;
}
ul>li{
 counter-increment: num;
 list-style: none;
}
ul>li:before{
 content: counters(num, "-") "、 ";
}
```

```
1、item1
2、item2
 2-1、item-sub1
 2-2、item-sub2
 2-3、item-sub3
3、item3
4、item4
```

- 选中元素第一行文本的第一个字母或者汉字

```
p:first-line {
 font-size: 36px;
 color: red;
 font-family: "Microsoft YaHei UI";
}
```

- 选中元素第一行文本

## 默认选中文本样式

言、年岁已高的导演们远离了舆论的漩

```
p::selection{
 color: red;
 background-color: yellow;
}
```

**注意：只能设置在块元素中**

年岁已高的导演们远离了舆论的漩涡，而取

设置了::selection后

- 控制打印机页面样式

```
@media print {
 .box{
 你的样式
 }
}
```

- attr() 获取元素属性值

```
@media print {
 a:before{
 content: attr(href);
 }
}
```



网页

[百度](http://www.baidu.com)

打印模式

<a href="http://www.baidu.com">百度</a>

[百度](http://www.baidu.com)

.box1

.box3  
是.box2的紧邻兄弟元素

.box1

.box2

.box3

.box4

.box5

.box2,.box3,.box4  
是.box2的兄弟元素

```
.box{
 width: 100px;
 height: 100px;
 background-color: #f0f0f0;
}
```

```
.box2+div{
 background-color: #00f;
}
```

```
<div class="box box1">.box1</div>
<div class="box box2">.box2</div>
<div class="box box3">.box3</div>
<div class="box box4">.box4</div>
<div class="box box5">.box5</div>
```

.box1

.box2

.box2是参照物

.box3

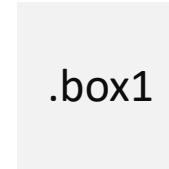
.box4

.box5

```
.box{
 width: 100px;
 height: 100px;
 background-color: #f0f0f0;
}
```

```
.box2~div{
 background-color: #f0f;
}
```

```
<div class="box box1">.box1</div>
<div class="box box2">.box2</div>
<div class="box box3">.box3</div>
<div class="box box4">.box4</div>
<div class="box box5">.box5</div>
```



.box2

.box2是参照物



.box3



.box4



.box5





HTML



# 快速开发 Emmet

讲师：许井龙

ngsteel@qq.com

- 一. Emmet 简介
- 二. Emmet 常用语法
- 三. 自定义 Emmet 模版

- Emmet的前身是大名鼎鼎的Zen coding。它使用仿CSS选择器的语法来生成代码，大大提高了HTML/CSS代码编写的速度。

官方网址：

<http://docs.emmet.io/abbreviations/syntax/>

- 子元素: div>ul>li
- 相邻元素: div+p+bq
- 层级提升: div+div>p>span+em^^^bq
- 相乘: ul>li\*5
- 分组: () div>(header>ul>li\*2>a)+footer>p
- ID and CLASS
  - div#header+div.page+div#footer.class1.class2.class3
- 自定义属性
  - td[title="Hello world!" colspan=3]
- 条目顺序号: \$
- 条目顺序号: ul>li.item\$\*5, ul>li.item\$\$\$\*5
- 改变条目计数起始值和方向
  - ul>li.item\$@-\*5 , ul>li.item\$@3\*5 , ul>li.item\$@-3\*5
- 文本: a{Click me}

WS Settings

live

## Editor &gt; Live Templates

By default expand with Tab

- select (<select name="..." id="...">...</select>)
- select+
- select:id (<select name="..." id="..." disabled>...</select>)
- select:disabled (<select name="..." id="..." disabled>...</select>)
- source:media (<source media="(...)" srcset="...">)
- source:media:sizes (<source media="(...)" sizes="..." srcset="...">)

Abbreviation: select Description: &lt;select name="..." id="..."&gt;...&lt;/select&gt;

Template text:

```
<select name="$VAR0$" id="$VAR1$">END

 <option name=""></option>

 <option name=""></option>

 <option name=""></option>

</select>
```

Expand with Default (Tab)

 Reformat according to styleApplicable in HTML: HTML Text; JavaScript: JSX HTML. [Change](#)