

一、介绍

(1) 浏览器：

前缀	浏览器
-webkit	goole chrom safari
-ms	IE
-moz	firefox
-o	oprea

(2)

预处理器：pre-processor ---less/sass(按照他们的规范写，把我们的代码生成可用的css代码去执行) cssNext插件(用来实现一些未来的标准的--未完全在各大浏览器上执行) postCss+cssNext

后处理器：post-processor ---autoprefixer插件 补齐代码，把我们的代码，按照规范去补齐兼容性，然后执行。往后的兼容性更好一些。postCss+autoprefixer
postCss----->用js实现的css的抽象的语法树--AST (Abstract Syntax Tree)，完成前半部分(语法树)，后半部分由各种插件(如：cssNext插件、autoprefixer插件)做。

二、css属性

1、 selector

(1) 关系选择器模式

E + F 下一个满足条件的兄弟元素节点F

E ~ F 下一个所有满足条件的兄弟元素节点F

E > F 直接子元素选择器F

(2) 属性选择器

E[attr~='val'];选择属性名为attr，属性值存在一个独立的val的元素

如：div[data~="a"]

E[attr|='val'] 选中以此值开头或者以此值加-开头的元素

如：div[class|'a']

E[attr^='val'] 选中属性名为attr，属性值以val开头的元素

如：div[class^='a']

E[attr\$='val'] 选中属性名为attr，属性值以val结尾的元素

E[attr*='val'] 选中属性名为attr，属性值里存在val的元素

(3) 伪元素选择器

E::placeholder 设置输入框的placeholder的样式

E::selection 改变字体选中以后的样子 兼容性很好

只能在里面设置三个属性：color background-color text-shadow--文字阴影

//伪类选择器 被选中的元素的一种状态

E:not(s)(attr) 选中没有attr的元素

E:root 和html相等，root--根标签选择器,一般写作:root

E:target 被标记成锚点之后 (location.hash = xxx(某结点的id值))

//考虑其他类型的元素对它的元素

E:first-child

E:last-child 不止看它这一类

E:only-child 选中独生的子元素E

E:nth-child(number/公式---n是自然数从0开始，公式中不能出现空格，css查数从1开始，奇数odd,偶数even)

E:nth-last-child 倒着查

E:first-of-type 是E这个类型的第一个孩子

E:last-of-type 是E这个类型的最后一个孩子

E:only-of-type 只有一个E这个类型的孩子

E:nth-of-type(number/公式) 对E这个类型的元素查

E:nth-of-last-type(number/公式)

E:empty 选中为空（没有节点）的E

E:checked 选中状态为checked的E元素

E:enabled 状态为enabled的E元素

E:disabled 状态为disabled的E元素

E:read-only 选中状态为readonly的元素

E:read-write 选中状态为readonly的元素

2、border

(1)border-radius

四个值，上右下左（顺时针），是水平和垂直方向的拉伸，border-radius的最大拉伸到div的宽高。

```
1 border-top-left-radius 10px 20px 水平和垂直方向的拉伸 先是水平半径 再垂直垂直半径
2 border-bottom-left-radius
3 border-radius:10px 20px 30px 40px / 10px 20px 30px 40px
```

(2)box-shadow

阴影大小基于边框原来位置向两边同时模糊 越大越虚

哪个阴影最先设置，就在最上面，不写时默认是外阴影 内阴影inset

box-shadow:(inset) 阴影水平偏移量 垂直偏移量 blur模糊值 spread阴影扩大，基于原来的宽高大小增加你设置的大小，四个方向同时增大 阴影颜色

例子：

```
1 .box{
2   width: 200px;
3   height: 200px;
4   position: absolute;
5   top: calc(50% - 150px);
6   left: calc(50% - 150px);
7   border-radius: 50%;
8   box-shadow: inset 0px 0px 50px #ffffff,
9   inset 10px 0px 80px #f0f,
10  inset -10px 0px 80px #0ff,
11  inset 10px 0px 300px #f0f,
12  inset -10px 0px 300px #0ff,
13  0px 0px 50px #ffffff,
14  -10px 0px 70px #f0f,
15  10px 0px 70px #0ff;
16  transition: all .6s;
17 }
```

效果：



(3)border-image

border-image:source slice repeat

- border-image-source
- border-image-slice :写数字和百分比，不能写像素，必须添，slice和border值一样，里面添加fill时，border设置成了背景色
- border-image-repeat:stretch--默认值/round(拉伸到一定程度就铺)/repeat(空白区域就铺)/space(空白到一定程度就铺)

//以下俩单独写出来

- border-image-outset : 背景图片往外延伸
- border-image-width: border里面能显示背景图片的宽度 默认值1，添auto时表示取slice的值加px

3、background

放多张背景图片时，调试background-size和 background-position。

(1)background-image

只能放到background-image里面的值：

- linear-gradient() 线性渐变： 线性的 to right/top/left/bottom / to top right /角度deg

如： 90deg ,#f0f 20px,#ff0 60px 颜色起止位置,20px之前是#f0f纯色，20~60是渐变的，60px以后#ff0纯色。

- radial-gradient() 径向渐变： 放射性的 ellipse(椭圆) (closest-corner) /circle(圆) at right/20%/10px bottom/20%/10px ,#f0f,green,#0 closest-corner/closest-side/farthest-corner /farthest-side

(2)background-origin : border-box/padding-box(默认值)/content-box，配合background-position使用 origin是谁，position就相对于谁定位。

(3) background-clip : 表示背景图片从哪儿截断不显示，border-box(默认值)/padding-box/content-box/text (用文字的部分截背景图片，除了文字体的区域展示背景图片，其他都不展示，只有在-webkit下好使 (-webkit-4/background-clip)，要配合-webkit-text-fill-color:transparent使用,此时文字变成了背景的一部分，文字阴影在背景的上面)。

(4)background-attachment : 改变图片的定位属性,scroll(相对于容器定位)/local(相对于内容区定位)/fixed(相对于视口，图片永远不动)。

(5)background-size:背景图片的尺寸。

都是用一张图片填充背景，让图片的一条边和容器的一条边对齐，然后1改变另外一条边的比例

cover：一定会让一张图片完整的填满容器，而不改变图片的原始比例，存在超出的风险，一条边对齐，另外一条边大于等于容器大小，不会出现两张图片

contain：不改变原有图片的原有比例，让容器包含一张完整的图片，存在repeat的风险，一条边对齐，另外一条边小于等于容器大小

如：div 500 700

img 100 200 ---> 500 1000一张图片完整的填充了cover

--->350 700 会出现repeat 一条边对齐，另外一条边小于等于容器大小

(6)background-repeat: 背景图片如何填充 :no-repeat/repeat-x/repeat-y/round/space

4、text

(1)text-shadow:水平偏移量x 垂直偏移量y 模糊值blur 颜色

```
1 .wrapper{
2   width: 100px;
3   height: 50px;
4   line-height: 50px;
5   position: absolute;
6   top: 100px;
7   left: 200px;
8   text-shadow: 10px 10px 2px #ff5000;
9 }
```

css3
css3

(2)-webkit-text-stroke描边：粗细 颜色 color:transparent 时，效果是镂空的。

拓展：@font-face{

font-family:给字体包起的名字，如'abc'

src:url(文字格式) format(格式)

}

format() 字体格式提示器 加强浏览器对字体的识别

字体的格式:trueuetype 微软和苹果共同研发的字体格式 .ttf 引用到苹果和微软电脑

opentype 微软和abode .opt

woff

.woff 兼容性有点问题

.eot ie支持的

.svg h5

MIME (.ttf .txt .pdf) 协议 定义了一些映射 浏览器-->操作系统

(3) white-space: nowrap (不换行) / normal / break-all

(4) word-break: break-word (尽可能的保持英文单词的完整性) / keep-all (不换行) / break-all (到边界 就换行，但英文单词会强制换行)

break-all:

做一个示例让大家更好的区分word-break: break-all; 的实际应用效果

```
.test1 {
  word-break: break-all;
}
```

break-all 会在文本内容遇见边界时，强制将文本打断换行，而不考虑单词是否是完整的一个单位

break-word:

做一个示例让大家更好的区分word-break: break-word; 的实际应用效果

```
.test2 {
  word-break: break-word;
}
```

break-word 同样也会在文本内容遇见边界时，强制将文本打断换行，不同的在于它会考虑单词是否完整，如果当前行无法放下需要被打断的单词，为了保持完整性，会将整个单词放到下一行进行展示

(5) word-wrap: break-word

(6) text-align

text-align: start ^{CSS3} | end ^{CSS3} | left | right | center | justify ^{CSS3} | match-parent ^{CSS3} | justify-all ^{CSS3}

默认值: start

适用于: 块容器

继承性: 有

动画性: 否

计算值: 指定值，除 match-parent 值外

媒体: 视觉

取值:

left: 内容左对齐。

center: 内容居中对齐。

right: 内容右对齐。

justify: 内容两端对齐，但对于强制打断的行（被打断的这一行）及最后一行（包括仅有一行文本的情况，因为它既是第一行也是最后一行）不做处理。（CSS3）

start: 内容对齐开始边界。（CSS3）

end: 内容对齐结束边界。（CSS3）

match-parent: 这个值和 [inherit] 表现一致，只是该值继承的 [start] 或 [end] 关键字是针对父元素的 'direction' 值并计算的，计算值可以是 left 和 right。（CSS3）

justify-all: 效果等同于 [justify]，不同的是最后一行也会两端对齐。（CSS3）

(7) text-align-last 处理文字的最后一行

(8) word-spacing: 单词间隔

(9) text-indent : 缩进

(10) vertical-align

vertical-align: baseline | sub | super | top | text-top | middle | bottom | text-bottom | **<percentage>** | **<length>**

默认值: **baseline**

适用于: 内联级与 table-cell 元素

继承性: 无

动画性: 当值为 **<length>** 时

计算值: 指定值

媒 体: 视觉

取值:

baseline: 把当前盒的基线与父级盒的基线对齐。如果该盒没有基线，就将底部外边距的边界和父级的基线对齐

sub: 把当前盒的基线降低到合适的位置作为父级盒的下标（该值不影响该元素文本的字体大小）

super: 把当前盒的基线提升到合适的位置作为父级盒的上标（该值不影响该元素文本的字体大小）

text-top: 把当前盒的top和父级的内容区的top对齐

text-bottom: 把当前盒的bottom和父级的内容区的bottom对齐

middle: 把当前盒的垂直中心和父级盒的基线加上父级的半x-height对齐

top: 把当前盒的top与行盒的top对齐

bottom: 把当前盒的bottom与行盒的bottom对齐

<percentage>: 把当前盒提升（正值）或者降低（负值）这个距离，百分比相对line-height计算。当值为0%时等同于baseline。

<length>: 把当前盒提升（正值）或者降低（负值）这个距离。当值为0时等同于baseline。（CSS2）

(11) text-size-adjust : 定义移动端页面中元素文本的大小如何调整，只在移动设备上生效。如果你的页面没有定义meta viewport，此属性定义将无效；

如果不希望页面的文本大小随手持设备尺寸变化（比如横竖屏旋转）而发生变化（这可能会导致页面布局错乱），可以定义值为none或者100%（早期版本的 Safari 会忽略none 取值）

text-size-adjust 分享到 复制本页链接

Base Browsers: IE8.0+, Firefox40.0+, Chrome40.0+, iOS8.0+, Android4.4+, Opera40.0+

语法:

text-size-adjust: auto | none | **<percentage>**

默认值: **auto**

适用于: 所有元素

继承性: 有

动画性: 当取值为 **<percentage>** 时

计算值: 指定值

媒 体: 视觉

取值:

auto: 文本大小根据设备尺寸进行调整。

none: 文本大小不会根据设备尺寸进行调整。

<percentage>: 用百分比来指定文本大小在设备尺寸不同的情况下如何调整。

(12) columns :column-width column-count 不适合瀑布流布局

- column-width:不能自适应列布局 多数柱高
- column-gap:间隙, ie是16px

- column-rule:和border一样
- column-fill
- column-span 贯穿整个列
- column-break-before :always,总在元素之前断裂产生新列
- column-break-after

举例：

```
1 .content{
2   column-width: 400px;
3   column-count: 3;
4   column-gap: 20px;
5   transform: translateX(0px);
6 }
```

人是会变的。人是会变的，不但人会变，世界上没有一样事物、没有一个生命是不变的，日日在变，秒秒在变。世上没有不变的东西，连喜马拉雅山和太平洋都在变，连地球、太阳都在变，何况是人。人会变得多，也会变得少；会变得好，也会变得坏。总之，人是一定会变的，不管这个人自己是不是想变，他一定会变。所以，当有人说“我不变，不会变”的时候，不管他的表情如何诚挚，不管他如何捶胸顿足，如何呼天抢地，如何信誓旦旦，如何赌咒发誓，他都是在说空话。相信了，是听的人的错和笨。人有权变。人有权变，有权觉得今是而昨非，为了种种原因：可告人或不可告人，聪明的或愚蠢的，有利可图的或蒙受损失的，各种不同的因素都可以变。变是变的人自己的事，每个人都有变的权利，把过去的自己完全推翻，别人看了虽觉气不顺，也只好干瞪眼。有一份体面的工作，且轻松舒服，收入还高，这是当今不少年轻人羡慕和追求的工作生活状态。有的人甚至把它作为人生的终极目标，到处炫耀。最近在网看到一则评论，题目：时代抛弃你，连一声招呼都不会打。前段时间，市值1867亿美元，全球最大的企业软件公司甲骨文，因为裁员而站在了风口浪尖。按理来说，一个公司内部裁员不该有这么大的动静，可一裁就是900人，而且都是全国各大名校毕业的精英，在被裁的中国研发区员工中，绝大多数都是37岁左右的工程师，年薪几十万甚至上百万。但如今，他们却只能拿着横幅，在街头喊着口号：“我们要工作！”“孩子

大型的专场招聘。可是，甲骨文员工的表现却让人大跌眼镜，大部分的人都未能通过第一轮的面试，有的即便过了面试，也在技术检测时，被刷了下来。可要知道，甲骨文对研发人员的招聘门槛相当高：“必须在清华、北大、上交、复旦和北邮，这五所学校读过本科才有资格进，而同时又必须具备硕士学历。”但恰恰是这些名校毕业的学生，却表现出了能力欠缺的一面。通过这两件事，我们似乎明白了一个道理：即便你曾经是一只战斗力极强的青蛙，在温水里面煮久了，也会慢慢死去。时代抛弃你，连声招呼都不会打。（悦读文网 www.yueduwen.com）

“不同的职业，对社会的贡献是不一样的。”当你选择了接受高学历教育的时候，是不是应该为这个社会做出相应的贡献呢？反观另一件事，阿里巴巴曾以40万年薪招聘资深体验师，看看它们的要求：“60岁以上老人，广场舞kol（意见领袖）优先。”当时很多人看到这个招聘广告的时候，很诧异，心想：“年过半百的老人，凭什么值那么多钱。”可令人意外的是，阿里一共收到3000多份简历，最后应聘成功的有：83岁的清华学霸奶奶，62岁的作家大爷，62岁IT大爷.....其中，83岁的清华学霸奶奶是十几个广场舞群的KOL，经常组织一些线下活动。62岁的IT大爷还带了自己做的ppt介绍自己：“12年淘宝买家经验、芝麻信用785分、熟练操作Photoshop设计软件.....”一边是，20几岁的硕士应聘环卫工“养老”，一边是，83岁清华奶奶应聘阿里“拼搏”，他们的差距究竟

力，没人会陪你原地停留。只有把命运掌握在自己手中，从今天起开始努力，即使暂时看不到希望，也要相信自己。因为比你牛几倍的人，依然在努力。不是每个人都能成为自己想要的样子，但每个人都可以努力成为自己想要的样子，因为世界上最可怕的两个词：一个叫执着，一个叫认真。只要在路上，就没有到不了的地方。奋斗的人生最精彩。人生每一天都是直播，人生不可能重来，更没有后悔药可吃，错过了就错过了。我们每个人都应该明白这样一个道理：“年轻时不吃苦，老来必受苦。”吃得苦中苦，方为人上人。苦和累是生活原味，只有吃苦才能吃香，要想人前显贵，必须人后受罪，没有人随随便便可以成功，那些成功人士马云、任正非等，哪一个不是吃苦吃出来的？我们应把“吃苦是财富、安逸是地狱、勤奋是捷径”当成自己的座右铭，多吃苦、吃大苦，苦过累过才能品尝到生活的甘甜，人这一辈子能记住的永远是最苦的日子，苦乐中奋斗的人生才会更精彩。奋斗的人生最现实。人活在这个世界上，总该留下一些什么？当我们回首往事的时候，不应碌碌无为而悔恨。行胜于言，心动不如行动，唯有奋斗才能让人生少些遗憾、少些悲凉、少些无奈，至少可以做到问心无愧。竞争是残酷的，生活是现实的。奋斗需要付出辛勤的汗水，天上不可能掉下馅饼，不劳而获是可耻的，投机取巧得来的不会长久。我们该怎样奋斗？也就是说在合适的年龄干合适的事。青少年时期就是好好学习，为人生打下坚实的基础；中青年时期就是好好

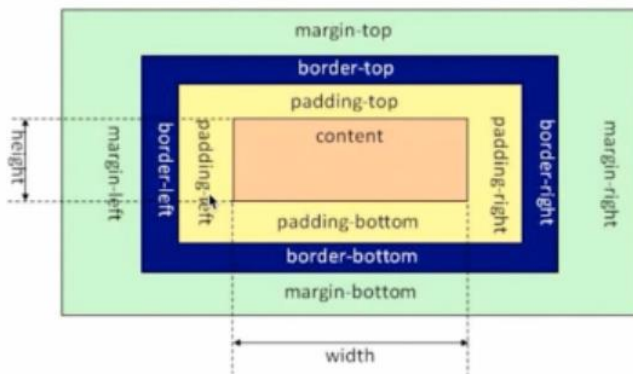
5、box

(1) 盒模型

标准模式 (w3c)：

w3c 关注内容宽高

$\text{boxWidth} = \text{width} + \text{border} * 2 + \text{padding} * 2$



标准模型

IE浏览器的怪异模式：

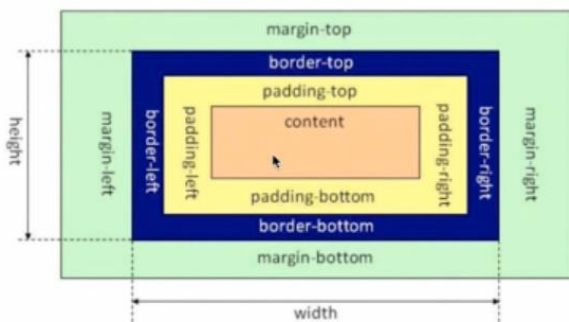
关注整体宽高

$\text{boxWidth} = \text{width}$

$\text{contentWidth} = \text{width} - \text{border} * 2 - \text{padding} * 2$

box-sizing: border-box 触发IE6混杂模式的盒模型 ---> 宽度不固定，但是内边距

固定



IE模型

css3设置两种模型：

```
1  /* 标准模型 */
2  box-sizing: content-box;
3
4  /* IE模型 */
5  box-sizing: border-box;
```

- overflow: hidden/visible (默认值) /scroll/auto (按需出现滚动条)

textarea里面的overflow的默认值是auto

overflow上当overflow-x或者overflow-y之一设置为非visible的值时，另一个属性会自动设置为auto

取值：

visible: 对溢出内容不做处理，内容可能会超出容器。

hidden: 隐藏溢出容器的内容且不出现滚动条。

scroll: 隐藏溢出容器的内容，溢出的内容可以通过滚动呈现。

auto: 当内容没有溢出容器时不出现滚动条，当内容溢出容器时出现滚动条，按需出现滚动条。`textarea` 元素的 `'overflow'` 默认值就是 `「auto」`。

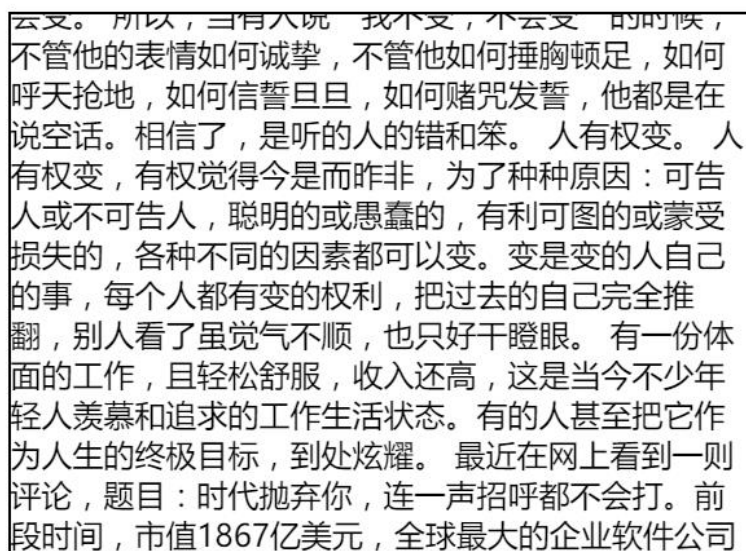
clip: 与 `「hidden」` 一样，`「clip」` 也被用来隐藏溢出容器的内容且不出现滚动条。不同的地方在于，`「clip」` 是一个完全禁止滚动的容器，而 `「hidden」` 仍然可以通过编程机制让内容可以滚动。

* CSS3 新增属性可能存在描述错误及变更，仅供参考，持续更新

- 当该级元素定义了 `'overflow'` 属性（包括 `'overflow-x'` 与 `'overflow-y'`）值为 `「非 visible」` 时，将会为它的内容创建一个新的块格式化上下文（BFC）。
- 对于 `table` 元素来说，假如其 `'table-layout'` 属性设置为 `「fixed」`，则 `td`、`th` 元素支持将 `'overflow'` 设为 `「hidden」`、`「scroll」` 或 `「auto」`，此时超出单元格尺寸的内容将被剪切。如果设为 `「visible」`，将导致额外的文本溢出到右边或左边（视 `'direction'` 属性设置而定）的单元格。
- `'overflow-x'` 属性用于指定元素水平方向上的内容溢出时的处理方式，`'overflow-y'` 属性用于指定元素垂直方向上的内容溢出时的处理方式。
- 当 `'overflow-x'`、`'overflow-y'` 中任意一个属性值的定义为 `「非 visible」` 时，另一个属性会自动将默认值 `「visible」` 计算为 `「auto」`。

举例：

```
1 .wrapper{
2   width: 400px;
3   height: 300px;
4   border: 1px solid black;
5   position: absolute;
6   top: 100px;
7   left: 200px;
8   overflow: scroll;
9 }
```



云变。所以，当有人说“我不变，不会变”的时候，不管他的表情如何诚挚，不管他如何捶胸顿足，如何呼天抢地，如何信誓旦旦，如何赌咒发誓，他都是在说空话。相信了，是听的人的错和笨。人有权变。人有权变，有权觉得今是而昨非，为了种种原因：可告人或不可告人，聪明的或愚蠢的，有利可图的或蒙受损失的，各种不同的因素都可以变。变是变的人自己的事，每个人都有变的权利，把过去的自己完全推翻，别人看了虽觉气不顺，也只好干瞪眼。有一份体面的工作，且轻松舒服，收入还高，这是当今不少年轻人羡慕和追求的工作生活状态。有的人甚至把它作为人生的终极目标，到处炫耀。最近在网上看到一则评论，题目：时代抛弃你，连一声招呼都不会打。前段时间，市值1867亿美元，全球最大的企业软件公司

```
1 .wrapper{
2   width: 400px;
3   height: 300px;
4   border: 1px solid black;
5   position: absolute;
6   top: 100px;
7   left: 200px;
8   overflow: auto;//按需出现滚动条 效果与上图进行对比，文字内容多少不一样
9 }
```

人是会变的。人是会变的，不但人会变，世界上没有一样事物、没有一个生命是不变的，日日在变，秒秒在变。世上没有不变的东西，连喜马拉雅山和太平洋都在变，连地球、太阳都在变，何况是人。人会变得多，也会变得少；会变得好，也会变得坏。总之，人是一定会变的，不管这个人自己是不是想变，他一定会变。所以，当有人说“我不变，不会变”的时候，不管他的表情如何诚挚，不管他如何捶胸顿足，如何呼天抢地，如何信誓旦旦，如何赌咒发誓，他都是在说空话。相信了，是听的人的错和笨。人有权变。

- **resize:**用户可以调节元素大小,不能单独使用，必须和**overflow:hidden**一起使用
none/both/horizontal/vertical

```
1 .wrapper{
2   width: 400px;
3   height: 300px;
4   border: 1px solid black;
5   position: absolute;
6   top: 100px;
7   left: 200px;
8   overflow: hidden;
9   resize: both;
10 }
```

人是会变的。人是会变的，不但人会变，世界上没有一样事物、没有一个生命是不变的，日日在变，秒秒在变。世上没有不变的东西，连喜马拉雅山和太平洋都在变，连地球、太阳都在变，何况是人。人会变得多，也会变得少；会变得好，也会变得坏。总之，人是一定会变的，不管这个人自己是不是想变，他一定会变。所以，当有人说“我不变，不会变”的时候，不管他的表情如何诚挚，不管他如何捶胸顿足，如何呼天抢地，如何信誓旦旦，如何赌咒发誓，他都是在说空话。相信了，是听的人的错和笨。人有权变。

(2) flex弹性盒子

设置到父元素上的：

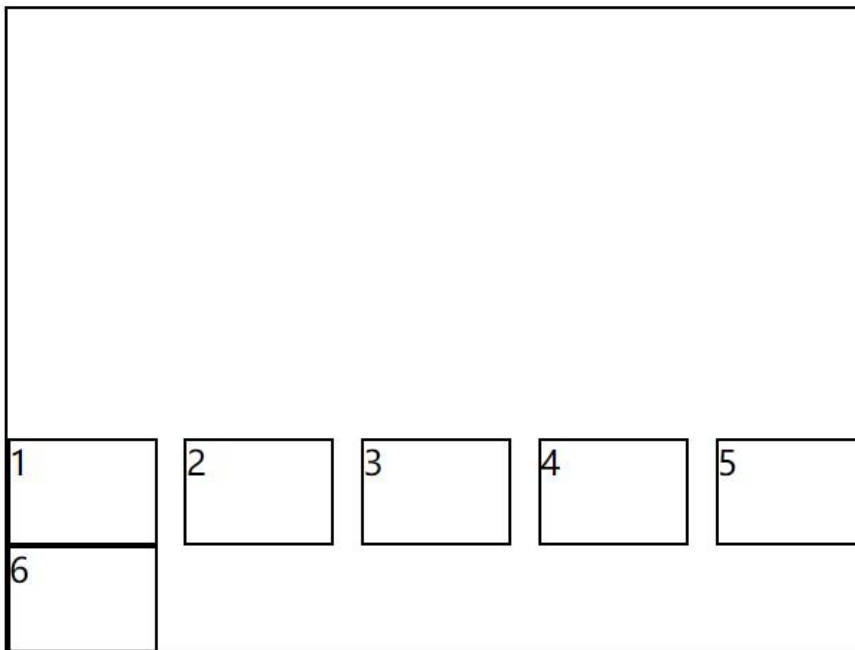
- display:flex/inline-flex
- flex-direction: 主轴方向（与交叉轴方向相反）row（水平）/column（垂直）/row-reverse/column-reverse（反向）
- flex-wrap:nowrap(不换行)/wrap(换行)/wrap-reverse(倒着换行)
- justify-content:基于主轴做一个对齐方式 flex-start/flex-end(基于主轴的在另外一个方向对齐)/flex-center(基于主轴在主轴的中间)/space-between(两边空格)/space-around(元素（两边）之间的空隙相等)
- align-content:基于交叉轴的位置分配，必须作用到多行元素上
- align-items:基于交叉轴的位置分配 stretch（子元素没设置高度时，给父元素设置这个，会使子元素自动撑开到父元素的高度,即沿主轴方向拉伸到父级的大小）/flex-start/flex-end/flex-center/baseline(基于文字的底线对齐)/center
-->主要针对单行元素来处理对齐方式

让一行元素全部水平垂直居中时，可以设置align-items:center;justify-content:center

举例子：

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>Title</title>
6 <style>
7 .wrapper{
8   width: 400px;
9   height: 300px;
10  border: 1px solid black;
11  position: absolute;
12  top: 100px;
```

```
13 left: 200px;
14 display: flex;
15 flex-direction: row; /*基于主轴方向横向排列*/
16 flex-wrap: wrap; /*超出了会换行，如果不加此代码，则不会超出，会压缩元素，让所有元素在一行*/
17 justify-content: space-between; /*元素之间会空格*/
18 align-content: flex-end; /*基于交叉轴的位置分配*/
19 }
20 .content{
21 width: 70px; /*一行元素加起来小于父元素的宽度时，justify-content: space-between才有效*/
22 height: 50px;
23 border: 1px solid black;
24 box-sizing: border-box;
25 }
26 </style>
27 </head>
28 <body>
29 <div class="wrapper">
30 <div class="content">1</div>
31 <div class="content">2</div>
32 <div class="content">3</div>
33 <div class="content">4</div>
34 <div class="content">5</div>
35 <div class="content">6</div>
36 </div>
37 </body>
38 </html>
```



设置到子元素上的：

- order: 排列 谁添的小谁在前面，order默认值为0，逻辑上在小的的上一层
- align-self:子元素作为一个个体，和交叉轴的对齐方式flex-start/flex-end，若父级再设置一个align-items，则自己的权重大，若给父级再设置一个align-content，则听父级的
- flex-grow:伸，默认值0，当这一行还有空间的时候，根据自己的比例让盒子伸张到一定的程度瓜分剩余部分
- flex-shrink:缩 按照加权值进行缩小,乘的是真实的内容区的宽高，默认值是1,0是不参与压缩，结合width或者flex-basis写

$100 \times 1 + 200 \times 1 + 400 \times 3 = 1500$ 真实内容区的大小*shrink值+....
400*3

----- *100px = 80 400-80=320px 100px--->多出来的值
1500

- flex-basis:，默认值auto,同时设置了这个和width时，它的优先级高，width的一个取代值。在不设置width只设置basis，或者basis > width 的时候，元素宽度代表元素的最小宽度值；width和basis一起设置，并且basis < width 时，basis表示元素宽度的下限，width表示元素宽度的上限.

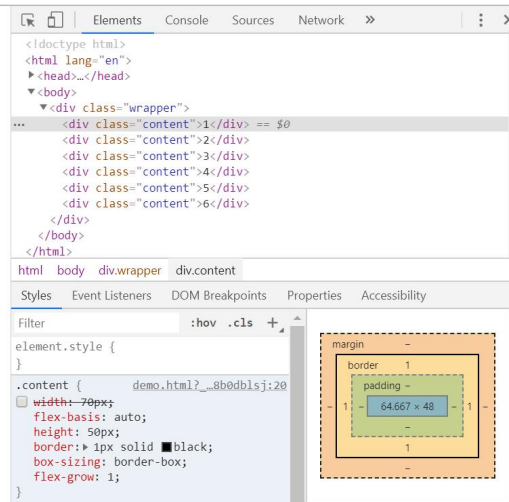
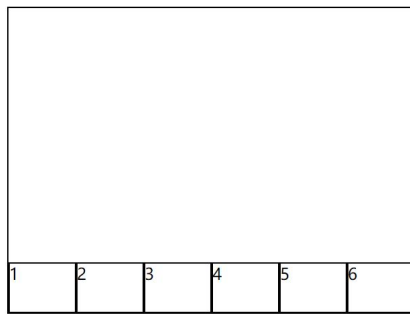
无论什么情况下，被不换行内容撑开的容器，不会被压缩计算，设置换行了才能参与正常压缩

举例：

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
```



```
4 <meta charset="UTF-8">
5 <title>Title</title>
6 <style>
7 .wrapper{
8 width: 400px;
9 height: 300px;
10 border: 1px solid black;
11 position: absolute;
12 top: 100px;
13 left: 200px;
14 display: flex;
15 flex-direction: row;/*基于主轴方向横向排列*/
16 flex-wrap: wrap;/*超出了会换行，如果不加此代码，则不会超出，会压缩元素，让所有元素在一行*/
17 justify-content: space-between;/*元素之间会空格*/
18 align-content: flex-end;/*基于交叉轴的位置分配*/
19 }
20 .content{
21 /*width: 70px;*/
22 flex-basis: auto;/*设置后的效果是，字体多大，就多宽*/
23 height: 50px;
24 border: 1px solid black;
25 box-sizing: border-box;
26 flex-grow: 1;/*默认值是0，不伸缩，1时表示当这一行还有空间的时候，根据自己的比例让盒子伸张到一定的程度瓜分剩余部分*/
27 }
28 </style>
29 </head>
30 <body>
31 <div class="wrapper">
32 <div class="content">1</div>
33 <div class="content">2</div>
34 <div class="content">3</div>
35 <div class="content">4</div>
36 <div class="content">5</div>
37 <div class="content">6</div>
38 </div>
39 </body>
40 </html>
```

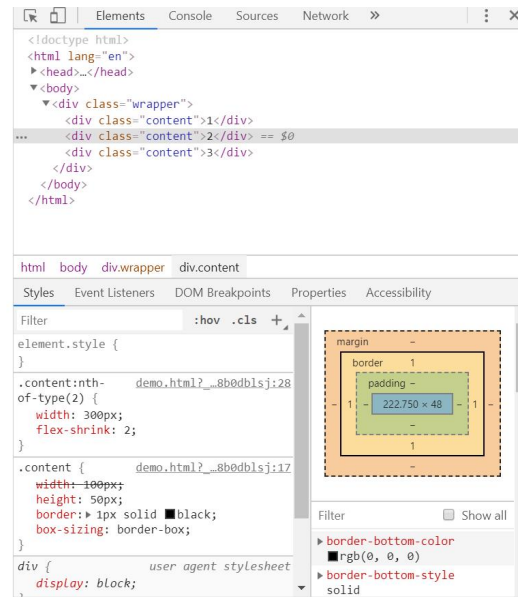
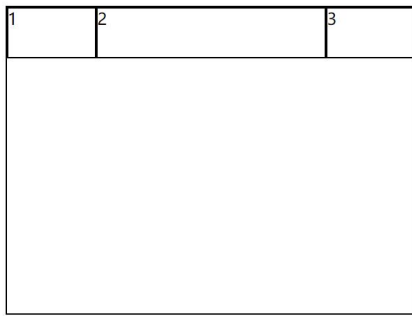


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Title</title>
6   <style>
7     .wrapper{
8       width: 400px;
9       height: 300px;
10      border: 1px solid black;
11      position: absolute;
12      top: 100px;
13      left: 200px;
14      display: flex;
15      flex-direction: row; /*基于主轴方向横向排列*/
16    }
17    .content{
18      width: 100px;
19      height: 50px;
20      border: 1px solid black;
21      box-sizing: border-box;
22    }
23    .content:nth-of-type(1){
24      flex-shrink: 1;
25    }
26    .content:nth-of-type(2){
27      width: 300px;
28      flex-shrink: 2;
29    }
```

```

30 .content:nth-of-type(3){
31   flex-shrink: 1;
32 }
33 </style>
34 </head>
35 <body>
36   <div class="wrapper">
37     <div class="content">1</div>
38     <div class="content">2</div>
39     <div class="content">3</div>
40   </div>
41 </body>
42 </html>

```



flex常见用途：

(1) 可动态增加导航栏

```

1 .wrapper{
2   width: 300px;
3   height: 200px;
4   border: 1px solid black;
5   display: flex;/*给父级添加*/
6 }
7 /*不用人为的改变宽度，无论加多少个都可以---->可动态增加导航栏*/
8 .item{ /*导航栏中的导航项,可动态添加*/
9   height: 30px;
10  line-height: 30px;
11  color: #fff;

```

```

12 font-size: 14px;
13 flex: 1 1 auto; /*关键*/ /*flex-grow flex-shrink flex-basis*/
14 text-align: center;
15 background-color: #09C369;
16 }
17
18
19 <div class="wrapper">
20   <div class="item">1</div>
21   <div class="item">2</div>
22   <div class="item">3</div>
23   <div class="item">4</div>
24 </div>

```



(2) 等分布局 , 4等分 , 2等分...中间可以加margin

```

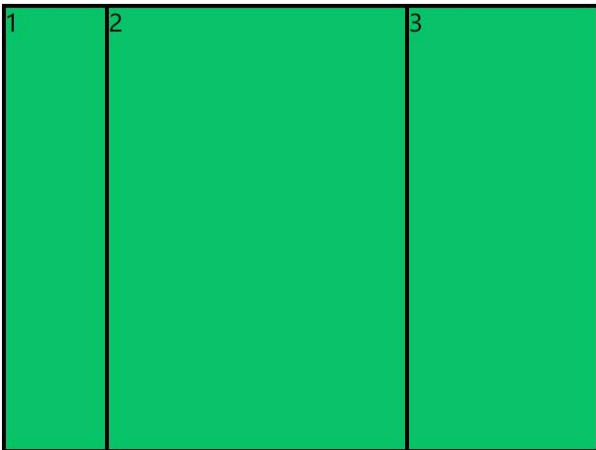
1 .wrapper1{
2   width: 400px;
3   height: 300px;
4   border: 1px solid black;
5   display: flex;
6 }
7 .content{
8   flex: 1 1 auto;
9   background-color: #09C369;
10  border: 1px solid black;

```

```

11 }
12 .content:nth-of-type(3){
13   flex: 2 2 auto;
14 }
15 .content:nth-of-type(2){
16   flex: 0 0 200px;
17 }

```



(3) 布局，以前用 float布局(float:left/right)会产生浮动，对周边元素有影响,现在可以用flex，父级：display:flex,align-items:center

(4) 圣杯模式布局

```

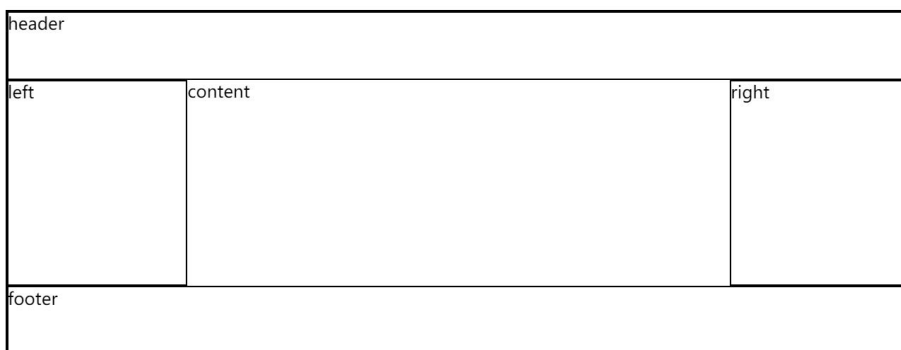
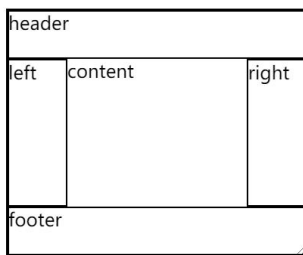
1  .wrapper2{
2    resize: both; /*只是为了作对比才写的代码*/
3    overflow: hidden;
4    width: 300px;
5    height: 300px;
6    border: 1px solid black;
7    display: flex;
8    flex-direction: column;
9  }
10 .header, .footer, .left, .right{
11   /*不参与伸缩，保持自身的固定宽度*/
12   flex: 0 0 20%;
13   border: 1px solid black;
14   box-sizing: border-box;
15 }
16 .contain{
17   flex: 1 1 auto;
18   display: flex;
19 }

```

```

20 .center{
21   flex: 1 1 auto;
22 }
23
24
25 <div class="wrapper2">
26   <div class="header">header</div>
27   <div class="contain">
28     <div class="left">left</div>
29     <div class="center">content</div>
30     <div class="right">right</div>
31   </div>
32   <div class="footer">footer</div>
33 </div>

```



6、transition:过渡动画

- transition-property : 变化的参数是什么 , all/width/height/.....默认值是 all
- transition-duration:运动秒数
- transition-timing-function :过渡动画的运动状态 , 默认值是ease平滑, cubic-bezier 贝塞尔曲线:linear,ease,

- transition-delay:等多少秒再过渡

如 : transition : width 2s linear 1s,height 1s

7、animation:多状态改变的动画

- animation-name
- animation-duration
- animation-timing-function
- animation-delay
- animation-iteration-count动画执行几次
- animation-direction走关键帧的方向 normal/reverse/alternate(正着走一次,倒着走一次,所以要求运动次数要大于等于2)/alternate-reverse
- animation-play-state:检索或设置运动状态 paused
- animation-fill-mode :让动画停留在某种状态none/forward(把元素变为运动之后的状态,在结束之后保留最后一帧的运动状态)/backwards(设置为动画开始之前的状态,在开始之前把元素设置为第一帧的状态)/both(设置元素状态为动画结束或开始的状态,等待运动时元素为第一帧状态,结束后元素为最后一帧状态)

```

1 @keyframes 兼容性--->-webkit-@keyframes
2 @keyframes run{
3   //关键帧 百分比定义运动顺序 百分数分割运动时间
4   0%{//from
5     left:0;
6     top:0;
7   }
8   25%{
9     left:100px;
10    top:0;
11  }
12   50%{
13     left:100px;
14     top:100px;
15   }
16   75%{
17     left:0;
18     top:100px;
19   }
20   100%{//to
21     left:0;
22     top:0;
23   }

```

举例：日出日落

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>sunrise</title>
6 <style>
7 *{
8   margin: 0;
9   padding: 0;
10 }
11 body{
12   background-color: black;
13 }
14 @keyframes skyChange {
15   0%{
16     opacity: 0.3;
17   }
18   25%{
19     opacity: 1;
20   }
21   50%{
22     opacity: 0.3;
23   }
24   75%{
25     opacity: 0.1;
26     /*background-color: ;*/
27   }
28   100%{
29     opacity: 0.3;
30     /*background-color: black;*/
31   }
32 }
33 .sky{
34   width: 100%;
35   height: 500px;
36   border: 1px solid black;
37   background-color: black;
```

```
38 background-image: linear-gradient(to bottom
, rgba(0,130,255,1), rgba(255,255,255,1));
39 animation: skyChange 10s cubic-bezier(.5,0,1,.5) infinite;
40 }
41 @keyframes sunChange {
42 0%{
43 opacity: 0;
44 /*transform: translateX(40px) translateY(0px);*/
45 }
46 10%{
47 opacity: 1;
48 transform: scale(.7,.7) translateX(0px) translateY(0px);
49 }
50 30%{
51 opacity: 1;
52 transform: scale(.5,.5) translateX(0px) translateY(-500px);
53 }
54 50%{
55 opacity: 0;
56 transform: scale(.7,.7) translateX(400px) translateY(0px);
57 }
58 100%{
59 opacity: 0;
60 /*transform: translateX(400px) translateY(200px);*/
61 }
62 }
63 .sun{
64 width: 100px;
65 height: 100px;
66 margin-left: 200px;
67 margin-top: 400px;
68 -webkit-border-radius: 50%;
69 -moz-border-radius: 50%;
70 border-radius: 50%;
71 transform: scale(.5,.5);
72 background-color:#ffffff;
73 box-shadow: 0px 0px 80px 30px #ffffff,
74 0px 0px 120px 50px #ff0;
75 animation: sunChange 10s infinite;
76 }
```

```
77 @keyframes moonChange {
78   0%{
79     opacity: 0;
80     transform: translateY(0) ;
81   }
82   50%{
83     opacity: 0;
84     transform: translateY(0) ;
85   }
86   70%{
87     opacity: 1;
88     transform: translateY(-250px);
89   }
90   80%{
91     opacity: 1;
92     transform: translateY(-250px);
93   }
94   90%{
95     opacity: 0;
96     transform: translateY(-250px);
97   }
98   100%{
99     opacity: 0;
100    transform: translateY(0);
101  }
102 }
103 .moon{
104   width: 100px;
105   height: 100px;
106   margin-right: 100px;
107   margin-top: -200px;
108   float: right;
109   position: relative;
110   border-radius: 50%;
111   background-color: #ffffff;
112   box-shadow: 0px 0px 16px #ffffff,
113   inset 0px 0px 5px #000000 ;
114   animation: moonChange 10s cubic-bezier(0,0,.5,.5) infinite;
115 }
116 .moon::after{
```

```

117 content: "";
118 position: absolute;
119 top: -12px;
120 left: -12px;
121 width: 90px;
122 height: 90px;
123 border-radius: 50%;
124 background-color: black;
125 /*box-shadow: 0px 0px 80px 50px #ffffff;*/
126 }
127 </style>
128 </head>
129 <body>
130 <div class="sky">
131 <div class="sun"></div>
132 <div class="moon"></div>
133 </div>
134 </body>
135 </html>

```

8、step

steps(1,start) 1----time

start 保留下一帧状态，直到这段动画时间结束

end 保留当前帧状态，直到这段动画时间结束，添一个forwards，常用。

steps(1,end);===step-end

steps(1,start);===step-start

举例：

```

1  /*打字*/
2  @keyframes cursor {
3    0%{
4      border-left-color: rgba(0,0,0,0);
5    }
6    50%{
7      border-left-color: rgba(0,0,0,1);
8    }
9    100%{
10     border-left-color: rgba(0,0,0,0);
11   }
12
13 }

```

```

14 @keyframes typeRun {
15   0%{
16     left: 0;
17   }
18   100%{
19     left: 100%;
20   }
21 }
22 .type{
23   position: relative;
24   display: inline-block;
25   height: 100px;
26   font-size: 80px;
27   line-height: 100px;
28   font-family: monospace;
29 }
30 div.type::after{
31   content: '';
32   position: absolute;
33   left: 0;
34   top: 10px;
35   height: 90px;
36   width: 100%;
37   background-color: #ffffff;
38   border-left: 2px solid black;
39   animation: cursor 1s steps(1,end) infinite,typeRun 14s steps(14,end) in
   finite;
40 }
41
42
43 <div class="type">abcdefjhijklmn</div>

```

9、rotate

rotate():2d变换

transform-origin : 旋转中心 center center 给谁设置，参照物就是谁

transform:rotateX() rotateY() rotateZ()

rotate3d(x,y,z,angle) x y z比值很重要，x y z形成一个矢量，元素以这个矢量为轴，旋转angle角度

10、scale:伸缩的是元素所在坐标轴的刻度

scale(x,y)

translateX(100px)视觉上平移了100px,实际是200px

scale(1,1)--->不变 多次写具有叠加操作效果 旋转带着伸缩的轴一起变换

saclex() sacley() saclez() 如同sacle(x,y,z)的拆分

sacle3d()类似于scale(x,y,z)

11、skew:倾斜的角度

skew(x,y) 倾斜的是坐标轴，而不是元素本身，并且坐标轴刻度还被拉伸了

skew(45deg,0) 倾斜的是y轴

skewx() skewy()

12、translate:将元素向指定的方向移动

transform:translate(-100px,0)实际上等于transform:translateX(-100px);

transform:translate(0,-100px)实际上等于transform:translateY(-100px)。

transform-origin:空间的变换中心

transform ; perspective(800px)--->加到子元素上的，多个元素景深效果一样，

元素居中显示：left:50%; top:50%; transform:translateX(-50%)-->元素的50%

13、perspective : 景深

设置到父级上，子级才有效果 结果基于元素在屏幕上的投影 可以看到不同元素在不同角度的位置---常用。 perspective-origin:center center --->默认值

当给一个元素设置了perspective或者transform-style时，该元素就变成了参照物

- transform-style:preserve-3d---元素创建三维空间 在元素的直接父级上加
- backface-visibility :hidden 让图片的后面隐藏

如：perspective:800px

transform:translateZ(800px)大到接近800的时候将看不到元素，---投影所致

举例：魔方

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>魔方</title>
6 <style>
7 @keyframes run {
8 0%{
9 transform: rotateY(0deg) rotateX(0deg);
10 }
11 100%{
12 transform: rotateY(135deg) rotateX(45deg);
13 }
14 }
```

```
15  }
16  .wrapper{
17  width: 800px;
18  height: 500px;
19  position: absolute;
20  top: 100px;
21  left: 300px;
22  /*perspective: 800px;*/
23  transform-style: preserve-3d;
24  animation: run 10s linear infinite;
25  }
26  .wrapper > div{
27  width: 300px;
28  height: 300px;
29  position: absolute;
30  }
31  .wrapper > div >img{
32  width: 100%;
33  height: 100%;
34  }
35  .box1{
36  transform: rotateY(90deg) translateZ(150px);
37  }
38  .box2{
39  transform: rotateY(-90deg) translateZ(150px);
40  }
41  .box3{
42  transform: rotateX(90deg) translateZ(150px);
43  }
44  .box4{
45  transform: rotateX(90deg) translateZ(150px);
46  }
47  .box5{
48  transform: translateZ(150px);
49  }
50  .box6{
51  transform: translateZ(-150px);
52  }
53  </style>
54  </head>
```

```
55 <body>
56   <div class="wrapper">
57     <div class="box1">
58       
59     </div>
60     <div class="box2">
61       
62     </div>
63     <div class="box3">
64       
65     </div>
66     <div class="box4">
67       
68     </div>
69     <div class="box5">
70       
71     </div>
72     <div class="box6">
73       
74     </div>
75   </div>
76 </body>
77 </html>
```

举例：3d旋转照片墙

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>猪小屁的照片墙</title>
6   <style>
7     *{
8       margin: 0;
9       padding: 0;
10    }
11    @keyframes run {
12      0%{
13        transform: rotateY(0deg);
14      }
15      100%{
16        transform: rotateY(360deg);
```

```
17  }
18  }
19  html,body{
20  width: 100%;
21  /*height: 800px;*/
22  }
23  .wrapper{
24  width: 100%;
25  perspective: 700px;
26  min-height: 100%;
27  perspective-origin: center 30%;
28  transform-style: preserve-3d;
29  position: relative;
30  z-index: 1000;
31  }
32  .content{
33  width: 160px;
34  position: absolute;
35  left: 50%;
36  top: 50%;
37  margin-top: 200px;
38  margin-left: -80px;
39  perspective: 800px;
40  transform-style: preserve-3d;
41  animation: run 10s linear infinite;
42  }
43  .content img{
44  border: 3px solid rgba(255,255,255,.5);
45  }
46  .content img{
47  max-width: 100%;
48  position: absolute;
49  }
50  .box{
51  width: 1000px;
52  height: 500px;
53  position: absolute;
54  }
55  .box > img{
56  width: 100%;
```

```
57 height: 500px;
58 position: absolute;
59 top: 200px;
60 left: 100px;
61 }
62 .content img:hover{
63 /*transform: scale(1.2,1.2);*/
64 }
65 .content a:nth-child(1) img{
66 transform: rotateY(0deg) translateZ(230px);
67 }
68 .content a:nth-child(2) img{
69 transform: rotateY(40deg) translateZ(230px);
70 }
71 .content a:nth-child(3) img{
72 transform: rotateY(80deg) translateZ(230px);
73 }
74 .content a:nth-child(4) img{
75 transform: rotateY(120deg) translateZ(230px);
76 }
77 .content a:nth-child(5) img{
78 transform: rotateY(160deg) translateZ(230px);
79 }
80 .content a:nth-child(6) img{
81 transform: rotateY(200deg) translateZ(230px);
82 }
83 .content a:nth-child(7) img{
84 transform: rotateY(240deg) translateZ(230px);
85 }
86 .content a:nth-child(8) img{
87 transform: rotateY(280deg) translateZ(230px);
88 }
89 .content a:nth-child(9) img{
90 transform: rotateY(320deg) translateZ(230px);
91 }
92 /*.content:hover{*/
93 /* animation-play-state: paused;*/
94 /*}*/
95 .box >img:not(:target){
96 z-index: -1;
```

```

97  }
98  </style>
99  </head>
100 <body>
101 <div class="box">
102 
103 
104 
105 
106 
107 
108 
109 
110 
111 </div>
112
113 <!-- -->
114 <div class="wrapper">
115 <div class="content">
116 <a href="#p1"></a>
117 <a href="#p2"></a>
118 <a href="#p3"></a>
119 <a href="#p4"></a>
120 <a href="#p5"></a>
121 <a href="#p6"></a>
122 <a href="#p7"></a>
123 <a href="#p8"></a>
124 <a href="#p9"></a>
125 <!-- <a href="#p10"></a>-->
126 </div>
127 </div>
128 </body>
129 </html>

```

14、matrix 矩阵计算

矩阵-----transform选中的计算规则

$$\begin{vmatrix} 1 & 0 & e \\ 0 & 1 & f \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} x+e \\ y+f \\ 1 \end{vmatrix}$$

经过矩阵计算后的结果就是元素变化的样子 2d平移了多少

$\text{matrix}(1,0,0,1,e,f) === \text{translate}(x,y)$

$\text{matrix}(a,0,0,d,0,0) === \text{scale}(x,y)$

15、显示屏和像素

空间混色法

rgb 光学三元色 红绿蓝

像素----三个像点构成-->红绿蓝--->空间混色法

像点

点矩 crt显示屏求点矩方法的意义，是几乎所有屏幕都通用的 屏幕点矩都不一样

crt显示屏-

lcd液晶屏

1920*1080 固定宽高下，展示的像素点数

像素---相对单位

物理像素 ---设备出厂时，像素的大小

dpi---一英寸所能容纳的像素点数

ppi----一英寸所能容纳的像素点数（点矩数）

参照像素 96dpi

设备像素比 dpr---物理像素/css像素

css编程--->逻辑像素 方式叫做逻辑屏幕 css:100px*100px,则若放到dpr=2物理像素上，
为200*200

根据不同的设备 逻辑屏转换为物理屏

按照psd的编程 美工用iphone6的标准给 按照ratino的物理像素去给我们图片的尺寸，要
除以dpr,如果是ios设备，要除以2(iphone6的dpr=2)

$\text{dpr} = 2, \text{img} = 200 * 200 \text{物理像素} \quad \text{css} = 100 * 100 < \text{----} 100 * \text{dpr}, 100 * \text{dpr}$

不看分辨率

1920*1080 ----要看dpi

10寸

2寸----清晰度大一些

360dpi/ppi *2.5英寸

$1920 / 2.5 \text{英寸} \text{---} = \text{dpi}$ (宽高除以英寸后的dpi是相同的，为设备dpi)

$1080 / \dots \text{英寸} \text{---} = \text{dpi}$

16、GPU性能问题

触发reflow: 避免reflow，节约性能

改变窗口大小

改变文字大小

内容改变，输入框输入文字

激活伪类，如: hover

操作class属性

脚本操作dom

计算offsetWidth offsetHeight

设置style属性

触发repaint:

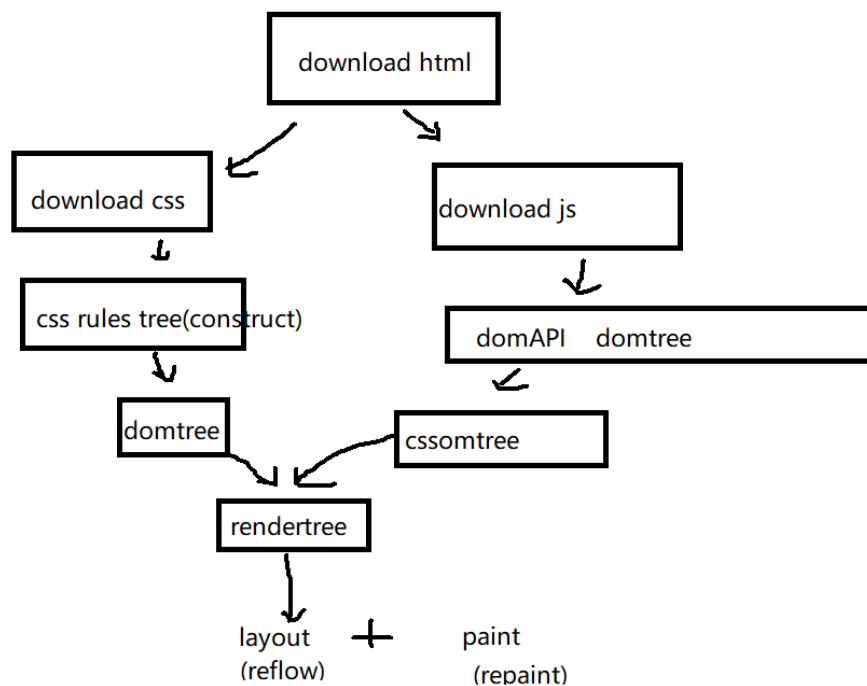
repaint: 如果只是改变某个元素的背景色、文字颜色、边框颜色，不影响它周围或内部布局的属性

repaint速度快于reflow

transform:...translatez(0) hack 提高性能

will-change: transform 标准的利用GPU加速处理性能

will-change: all 层数太多，消耗性能



逻辑图（多层矢量图）----->实际绘制（栅格化）

17、响应式网页开发

pc端--电脑 移动端---手机，平板 大小，分辨率

- 响应式网页设计或称自适应网页设计或称回应式网页设计/对应式网页设计，是一种网页设计的技术做法，该设计可使网站在不同的设备（从桌面计算机显示器到移动电话或其他移动产品设备）上浏览时对应不同分辨率皆有适合的呈现，减少用户进行缩放，平移和滚动等操作行为。

- 真正的响应式设计方法不仅仅是根据可视区域大小而改变网页布局，而是要从整体上颠覆当前网页的设计方法，是针对任意设备的网页内容进行完美布局的一种显示机制。

用一套代码解决几乎所有设备的页面展示问题

注：设计工作由产品经理或者美工来出

- 1css像素 并不完全等于 设备像素--根据屏幕分辨率进行相应的调整。Css像素根据设备像素进行计算，根据设备的分辨率 dpi值来计算css像素真正展现的大小

要适配各种不同分辨率的设备

- 模拟移动端的meta，如下代码：

```
1 <meta name="viewport" content="width=device-width,minimum-scale=1.0,maximum-scale=1.0,user-scalable=no">
2 width: 可视区宽度
3 device-width: 设备宽度
4 minimum-scale: 最小缩放比
5 maximum-scale: 最大缩放比
6 user-scalable: 是否允许用户缩放
```

响应式代码：

```
1 <!-- 将页面的大小根据分辨率不同进行相应的调节 以展示给用户的大小感觉上差不多-->
2 viewport默认视口(能看到的页面大小)大小 980px左右
3 content="视口宽度=设备宽度"
4 initial-scale初始化缩放比（针对视口而言，和content效果一样）
5 width = device-width:iphone或者ipad上，横竖屏的高度=竖屏时的宽度 不能自适应
6 initial-scale=1.0 :windows手机上 ie浏览器上横竖屏的宽度=竖屏时的宽度 不能自适应
7 user-scalable: 是否允许用户缩放 no
8 <!-- 适配各种不同分辨率的设备: -->
9 <meta name="viewport" content="width=device-width,initial-scale=1.0,user-scalable=no">
```

- 响应式网页的开发方法：

1. 流体网格：可伸缩的网格（大小宽高 都是可伸缩（可用flex或者百分比来控制大小）float）---》布局上面 元素大小不固定可伸缩

2. 弹性图片：图片宽高不固定（可设置min-width: 100%）

3. 媒体查询：让网页在不同的终端上面展示效果相同（用户体验相同-->让用户用着更爽）在不同的设备（大小不同 分辨率不同）上面均展示合适的页面

4. 主要断点：设备宽度的临界点

大小的区别 ---》 宽度不同 ---》 根据不同宽度展示不同的样式
响应式网页开发主要是在css样式上面进行操作

- 媒体查询

媒体查询是向不同设备提供不同样式的一种方式，它为每种类型的用户提供了最佳的体验。

css2: media typemedia type(媒体类型)是css 2中的一个非常有用的属性，通过media type我们可以对不同的设备指定特定的样式，从而实现更丰富的界面。

css3: media querymedia query是CSS3对media type的增强，事实上我们可以将media query看成是media type+css属性(媒体特性Media features)判断。

使用方法：

媒体类型 (Media Type): all(全部)、screen(屏幕)、print(页面打印或打印预览模式)

媒体类型

值	描述
all	用于所有设备
aural	已废弃。用于语音和声音合成器
braille	已废弃。应用于盲文触摸式反馈设备
embossed	已废弃。用于打印的盲人印刷设备
handheld	已废弃。用于掌上设备或更小的装置，如PDA和小型电话
print	用于打印机和打印预览
projection	已废弃。用于投影设备
screen	用于电脑屏幕，平板电脑，智能手机等。
speech	应用于屏幕阅读器等发声设备
tty	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
tv	已废弃。用于电视和网络电视

媒体特性 (Media features): width(渲染区宽度)、device-width(设备宽度)...

Media Query是CSS3 对Media Type的增强版，其实可以将Media Query看成Media Type(判断条件)+CSS(符合条件的样式规则)

媒体功能

值	描述
aspect-ratio	定义输出设备中的页面可见区域宽度与高度的比率
color	定义输出设备每一组彩色原件的个数。如果不是彩色设备，则值等于0
color-index	定义在输出设备的彩色查询表中的条目数。如果没有使用彩色查询表，则值等于0
device-aspect-ratio	定义输出设备的屏幕可见宽度与高度的比率。
device-height	定义输出设备的屏幕可见高度。
device-width	定义输出设备的屏幕可见宽度。
grid	用来查询输出设备是否使用栅格或点阵。
height	定义输出设备中的页面可见区域高度。
max-aspect-ratio	定义输出设备的屏幕可见宽度与高度的最大比率。
max-color	定义输出设备每一组彩色原件的最大个数。
max-color-index	定义在输出设备的彩色查询表中的最大条目数。
max-device-aspect-ratio	定义输出设备的屏幕可见宽度与高度的最大比率。
max-device-height	定义输出设备的屏幕可见的最大高度。
max-device-width	定义输出设备的屏幕最大可见宽度。
max-height	定义输出设备中的页面最大可见区域高度。
max-monochrome	定义在一个单色框架缓冲区中每像素包含的最大单色原件个数。
max-resolution	定义设备的最大分辨率。
max-width	定义输出设备中的页面最大可见区域宽度。
min-aspect-ratio	定义输出设备中的页面可见区域宽度与高度的最小比率。

媒体查询的引用方法有很多种：

1. link标签

```
1 <link rel="stylesheet" media="screen and (max-width:375px)" href="index.css">
```

2. @import url(example.css) screen and (width:800px);

```
1 @import url( index.css) screen and (width:800px);
2
3 @import url("global.css");
4 @import url(global.css);
5 @import "global.css";
```

3. css3新增的@media

```
1 <style>
2 @media(max-width:375px){
3   html,body{
```

```

4 width:100%;
5 height:100%
6 }
7 }----->css样式引入
8 </style>
9 //在style标签外
10 @media screen and (min-width: 600px) and (max-width:100px);

```

- 逻辑操作符

合并多个媒体属性 and

@media screen and (min-width: 600px) and (max-width:100px) ;

合并多个媒体属性或合并媒体属性与媒体类型, 一个基本的媒体查询, 即一个媒体属性与默认指定的screen媒体类型。

- 指定备用功能 or

```
1 @media screen and (min-width: 769px), print and (min-width: 6in)“
```

没有or关键词可用于指定备用的媒体功能。相反, 可以将备用功能以逗号分割列表的形式列出

这会将样式应用到宽度超过769像素的屏幕或使用至少6英寸宽的纸张的打印设备。

- 指定否定条件 not

```
1 @media not screen and (monochrome)
```

要指定否定条件, 可以在媒体声明中添加关键字not, 不能在单个条件前使用not。该关键字必须位于声明的开头, 而且它会否定整个声明。所以, 上面的示例会应用于除单色屏幕外的所有设备。

- 向早期浏览器隐藏媒体查询only

```
1 media="only screen and (min-width: 401px) and (max-width: 600px)"
```

媒体查询规范还提供了关键字only, 它用于向早期浏览器隐藏媒体查询。类似于not, 该关键字必须位于声明的开头。Only指定某种特定的媒体类型 为了兼容不支持媒体查询的浏览器

早期浏览器应该将以下语句media="screen and (min-width: 401px) and (max-width: 600px)"

解释为media="screen": 换句话说, 它应该将样式规则应用于所有屏幕设备, 即使它不知道媒体查询的含义。

无法识别媒体查询的浏览器要求获得逗号分割的媒体类型列表, 规范要求, 它们应该在第一个不是连字符的非数字字母字符之前截断每个值。所以, 早期浏览器应该将上面的示例解释为: media="only"

因为没有only这样的媒体类型, 所以样式表被忽略。

Query ---> css3

注：device-width/device-height 是设备的宽度（如电脑手机的宽度 不是浏览器的宽度）

width/height使用documentElement.clientWidth/Height即viewport的值。渲染宽度/高度

视口宽度

- 单位值

Rem：rem是CSS3新增的一个相对单位（root em，根em）相对的只是HTML根元素的字体大小。所以在此之前给html设置字体大小。--->结合媒体查询使用，在媒体查询里面设置html的字体大小

Em：em是相对长度单位。相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。

Px：px像素（Pixel）。相对长度单位。像素px是相对于显示器屏幕分辨率而言的。

Vw：相对于视口的宽度。视口被均分为100单位的vw

Vh：相对于视口的高度。视口被均分为100单位的vh

Vmax：相对于视口的宽度或高度中较大的那个。其中最大的那个被均分为100单位的vmax

Vmin：相对于视口的宽度或高度中较小的那个。其中最小的那个被均分为100单位的vmin

渐进增强 ---》 iphone6 向上兼容 兼容最新设备

优雅降级 ---》 开发通用版本 再兼容老版本 向下兼容

先移动端 ---》 pc端

先iphone6为初始原型 开发 ---》 兼容其他的设备 ===》 渐进增强

另：css3各属性兼容性：<http://c.biancheng.net/view/1281.html>