目录

[1 常用属性 3](#_Toc89559669)

[1.1 display的属性值及其作用 3](#_Toc89559670)

[1.2 link和@import的区别 4](#_Toc89559671)

[1.3 transition和animation的区别 4](#_Toc89559672)

[1.4 display:none与visibility:hidden的区别 4](#_Toc89559673)

[1.5 对line-height 的理解及其赋值方式 4](#_Toc89559674)

[1.6 ::before 和 :after 的双冒号和单冒号有什么区别 5](#_Toc89559675)

[1.7 Sass、Less 是什么 为什么要使用他们 6](#_Toc89559676)

[1.8 z-index属性在什么情况下会失效 6](#_Toc89559677)

[1.9 position属性 6](#_Toc89559678)

[1.10 Flex布局的理解及其使用场景 9](#_Toc89559679)

[1.11 display、float、position的关系 9](#_Toc89559680)

[1.12 absolute与fixed共同点与不同点 10](#_Toc89559681)

[1.13 对 sticky 定位的理解 10](#_Toc89559682)

[1.14 设置小于12px的字体 11](#_Toc89559683)

[1.15 png、jpg、gif 11](#_Toc89559684)

[1.16 overflow 11](#_Toc89559685)

[1.17 float属性 12](#_Toc89559686)

[2 常考元素布局方法 14](#_Toc89559687)

[2.1 常见的实现左右两栏布局的方法 14](#_Toc89559688)

[2.2 隐藏元素的方法有哪些 15](#_Toc89559689)

[2.3 对line-height 的理解 15](#_Toc89559690)

[2.4 判断元素是否到达可视区域 16](#_Toc89559691)

[2.5 常见的CSS布局单位 16](#_Toc89559692)

[2.6 如何根据设计稿进行移动端适配 17](#_Toc89559693)

[2.7 清除浮动 17](#_Toc89559694)

[2.8 元素的层叠顺序 18](#_Toc89559695)

[2.9 常见居中方案 19](#_Toc89559696)

[2.10 如何居中p 如何居中浮动元素 如何让绝对定位的p居中 20](#_Toc89559697)

[2.11 常见的inline元素、block元素、inline-block元素 20](#_Toc89559698)

[2.12 如何来管理所有CSS文件,js 与图片 21](#_Toc89559699)

[2.13 CSS auto 22](#_Toc89559700)

[3 CSS基础知识 22](#_Toc89559701)

[3.1css 的渲染层 22](#_Toc89559702)

[3.2 CSS三大特性——优先级 22](#_Toc89559703)

[3.3 CSS三大特性——继承级 24](#_Toc89559704)

[3.4 CSS三大特性——层叠级 25](#_Toc89559705)

[3.5 CSS中可继承与不可继承属性有哪些 25](#_Toc89559706)

[3.6 伪元素和伪类的区别和作用 27](#_Toc89559707)

[3.7 CSS盒模型 28](#_Toc89559708)

[3.8 CSS3中有哪些新特性 29](#_Toc89559709)

[3.9 CSS 优化和提高性能的方法有哪些 29](#_Toc89559710)

[3.10 CSS预处理器/后处理器是什么 30](#_Toc89559711)

[3.11 ::和 :的双冒号和单冒号有什么区别 30](#_Toc89559712)

[4 实战篇字节 31](#_Toc89559713)

[4.1 字节CSS实战 31](#_Toc89559714)

[4.1.1 用CSS实现一个三角形 31](#_Toc89559715)

# 1 常用属性

## 1.1 display的属性值及其作用

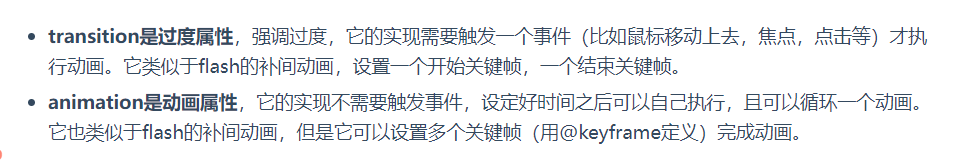


## 1.2 link和@import的区别

两者都是外部引用CSS的方式，它们的区别如下



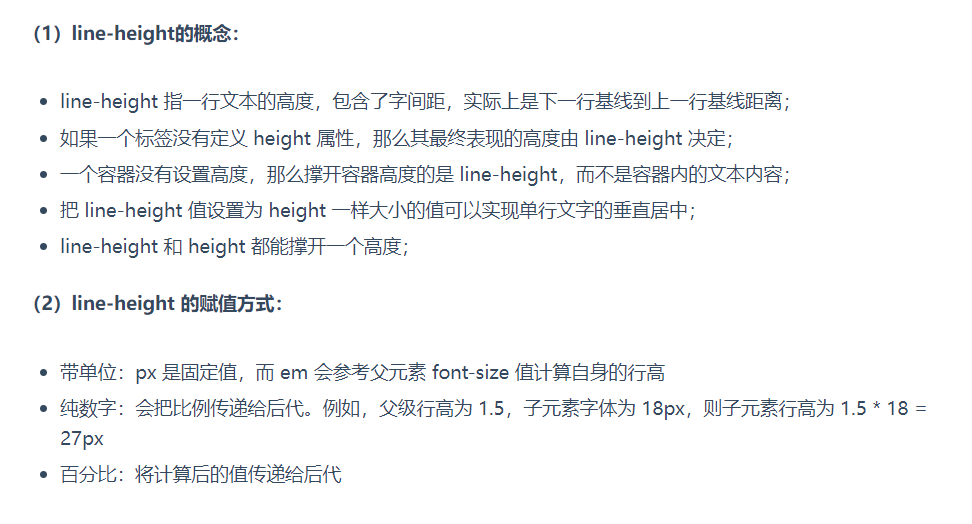
## 1.3 transition和animation的区别



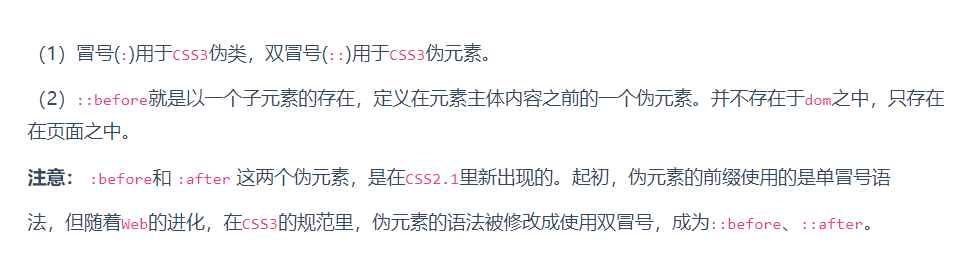
## 1.4 display:none与visibility:hidden的区别



## 1.5 对line-height 的理解及其赋值方式



## 1.6 ::before 和 :after 的双冒号和单冒号有什么区别



**伪元素:before和:after添加的内容默认是inline元素；**这个两个伪元素的content属性，表示伪元素的内容,设置:before和:after时必须设置其content属性，否则伪元素就不起作用。

这个两个伪元素在真正页面*元素内部*之前和之后添加新内容（当然了，可以对伪元素应用定位可以置于任何位置）。可以用以下例子来说明：

<p>wonyun!</p>

<style>

p:before{content: "hello "}

p:after{content: "you are handsome!"}

</style>

上面例子从技术角度看，等价于下面的html结构：

<p>

<span>hello </span>

wonyun!

<span> you are handsome!</span>

</p>

## 1.7 Sass、Less 是什么 为什么要使用他们

他们都是 CSS 预处理器，是 CSS 上的一种抽象层。他们是一种特殊的语法/语言编译成 CSS。例如 Less 是一种动态样式语言，将 CSS 赋予了动态语言的特性，如变量，继承，运算， 函数，LESS 既可以在客户端上运行 (支持 IE6+, Webkit, Firefox)，也可以在服务端运行 (借助 Node.js)。

为什么使用？

结构清晰，便于扩展。 可以方便地屏蔽浏览器私有语法差异。封装对浏览器语法差异的重复处理， 减少无意义的机械劳动。可以轻松实现多重继承。 完全兼容 CSS 代码，可以方便地应用到老项目中。LESS 只是在 CSS 语法上做了扩展，所以老的 CSS 代码也可以与 LESS 代码一同编译。

## 1.8 z-index属性在什么情况下会失效

通常 z-index 的使用是在有两个重叠的标签，在一定的情况下控制其中一个在另一个的上方或者下方出现。z-index值越大就越是在上层。z-index元素的position属性需要是relative，absolute或是fixed。z-index属性在下列情况下会失效。

·父元素position为relative时，子元素的z-index失效。解决：父元素position改为absolute或static。

·元素没有设置position属性为非static属性。解决：设置该元素的position属性为relative，absolute或是fixed中的一种。

## 1.9 position属性

①**absolute** ：绝对定位；脱离文档流的布局，遗留下来的空间由后面的元素填充。定位的起始位置为最近的父元素(postion不为static)，否则为Body文档本身。

②**relative** ：相对定位；不脱离文档流的布局，只改变自身的位置，在文档流原先的位置遗留空白区域。定位的起始位置为此元素原先在文档流的位置。

③**fixed** ：固定定位；类似于absolute，但不随着滚动条的移动而改变位置。

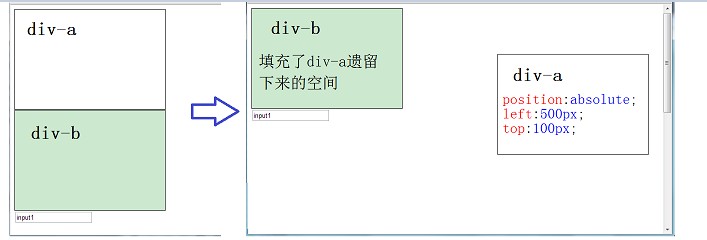
④**static** ：默认值；默认布局。

2.1 position:absolute

2.1.1 说明

**绝对定位**；脱离文档流的布局，遗留下来的空间由后面的元素填充。定位的起始位置为最近的父元素(postion不为static)，否则为Body文档本身。

2.1.2 视图

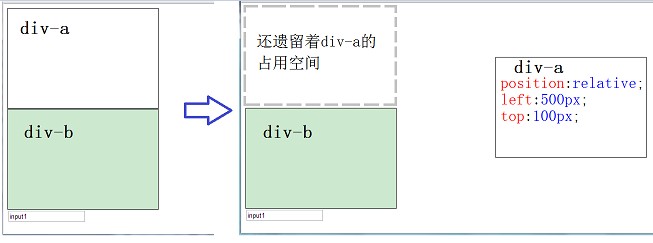


2.2 position:relative

2.2.1 说明

**相对定位**；不脱离文档流的布局，只改变自身的位置，在文档流原先的位置遗留空白区域。定位的起始位置为此元素原先在文档流的位置。

2.2.2 视图

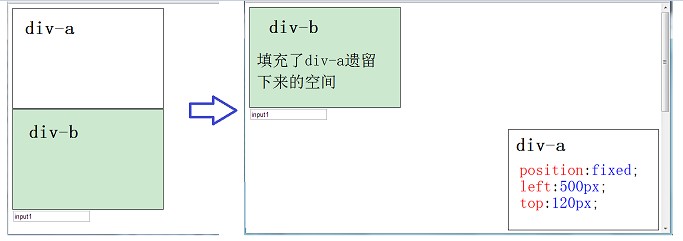


2.3 position:fixed

2.3.1 说明

**固定定位**；类似于absolute，但不随着滚动条的移动而改变位置。

2.3.2 视图



2.3.3 应用场景

①登录框覆盖层：如dz论坛的登录。

②虚假QQ消息广告。

2.4 position:static

2.4.1 说明

**默认定位**，表示此元素为默认定位方式。

## 1.10 Flex布局的理解及其使用场景

Flex是Flexible Box的缩写，意为”弹性布局”，用来为盒状模型提供最大的灵活性。

任何一个容器都可以指定为Flex布局。

采用Flex布局的元素，称为Flex容器（flex container），简称”容器”。它的所有子元素自动成为容器成员，称为Flex项目（flex item），简称”项目”。



容器默认存在两根轴：水平的主轴（main axis）和垂直的交叉轴（cross axis）。主轴的开始位置（与边框的交叉点）叫做main start，结束位置叫做main end；交叉轴的开始位置叫做cross start，结束位置叫做cross end。

项目默认沿主轴排列。单个项目占据的主轴空间叫做main size，占据的交叉轴空间叫做cross size。

justify-content属性定义了项目在主轴上的对齐方式

align-items属性定义项目在交叉轴上如何对齐

align-content属性定义了多根轴线的对齐方式。如果项目只有一根轴线，该属性不起作用。

## 1.11 display、float、position的关系



## 1.12 absolute与fixed共同点与不同点

**共同点**

改变行内元素的呈现方式，将display置为inline-block。

使元素脱离普通文档流，不再占据文档物理空间。

**不同点**

abuselute与fixed的根元素不同，abuselute的根元素可以设置，fixed根元素是浏览器。

在有滚动条的页面中，absolute会跟着父元素进行移动，fixed固定在页面的具体位置。

## 1.13 对 sticky 定位的理解

sticky 英文字面意思是粘贴，所以可以把它称之为粘性定位。语法：**position: sticky;** 基于用户的滚动位置来定位。

粘性定位的元素是依赖于用户的滚动，在 **position:relative** 与 **position:fixed** 定位之间切换。它的行为就像**position:relative;** 而当页面滚动超出目标区域时，它的表现就像 **position:fixed;**，它会固定在目标位置。元素定位表现为在跨越特定阈值前为相对定位，之后为固定定位。这个特定阈值指的是 top, right, bottom 或 left 之一，换言之，指定 top, right, bottom 或 left 四个阈值其中之一，才可使粘性定位生效。否则其行为与相对定位相同。

## 1.14 设置小于12px的字体

在谷歌下css设置字体大小为12px及以下时，显示都是一样大小，都是默认12px。

使用Webkit的内核的-webkit-text-size-adjust的私有CSS属性来解决，只要加了-webkit-text-size-adjust:none;字体大小就不受限制了。

使用css3的transform缩放属性-webkit-transform:scale(0.5)

## 1.15 png、jpg、gif

png是便携式网络图片（Portable Network Graphics）是一种无损数据压缩位图文件格式.优点是：压缩比高，色彩好。 大多数地方都可以用。

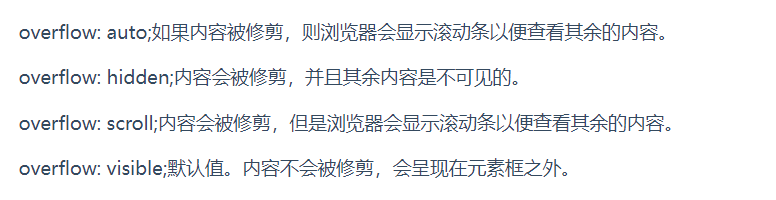
jpg是一种针对相片使用的一种失真压缩方法，是一种破坏性的压缩，在色调及颜色平滑变化做的不错。在www上，被用来储存和传输照片的格式。

gif是一种位图文件格式，以8位色重现真色彩的图像。可以实现动画效果。

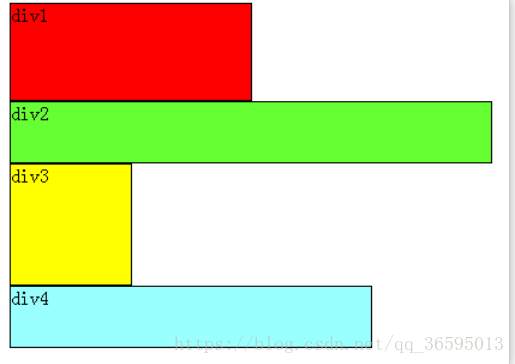
## 1.16 overflow

overflow 属性规定当内容溢出元素框时发生的事情

这个属性定义溢出元素内容区的内容会如何处理。如果值为 scroll，不论是否需要，用户代理都会提供一种滚动机制。因此，有可能即使元素框中可以放下所有内容也会出现滚动条

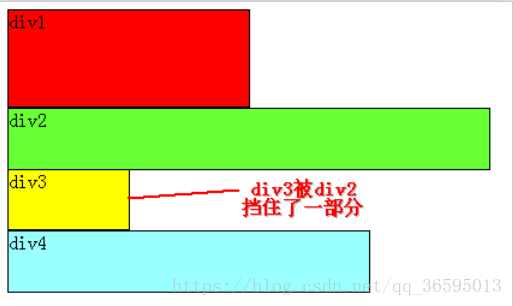


## 1.17 float属性

首先要知道，div是块级元素，在页面中独占一行，自上而下排列，也就是传说中的流。如下图：  
  
可以看出，即使div1的宽度很小，页面中一行可以容下div1和div2，div2也不会排在div1后边，因为div元素是独占一行的。

注意，以上这些理论，是指标准流中的div。  
小菜认为，无论多么复杂的布局，其基本出发点均是：**“如何在一行显示多个div元素”。**  
显然标准流已经无法满足需求，这就要用到浮动。

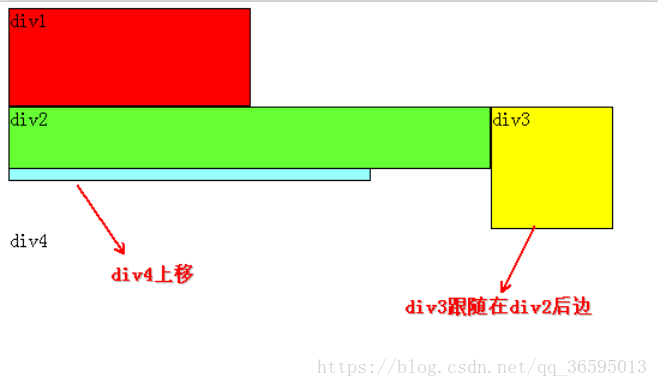
**浮动可以理解为让某个div元素脱离标准流，漂浮在标准流之上，和标准流不是一个层次。**

例如，假设上图中的div2浮动，那么它将脱离标准流，但div1、div3、div4仍然在标准流当中，所以div3会自动向上移动，占据div2的位置，重新组成一个流。如图：  
  
从图中可以看出，由于对div2设置浮动，因此它不再属于标准流，div3自动上移顶替div2的位置，div1、div3、div4依次排列，成为一个新的流。又因为浮动是漂浮在标准流之上的，因此div2挡住了一部分div3，div3看起来变“矮”了。

这里div2用的是左浮动(float:left;)，可以理解为漂浮起来后靠左排列，右浮动(float:right;)当然就是靠右排列。这里的靠左、靠右是说页面的左、右边缘。

如果我们把div2采用右浮动，会是如下效果：  
  
此时div2靠页面右边缘排列，不再遮挡div3，读者可以清晰的看到上面所讲的div1、div3、div4组成的流。

目前为止我们只浮动了一个div元素，多个呢？

下面我们把div2和div3都加上左浮动，效果如图：  
  
同理，由于div2、div3浮动，它们不再属于标准流，因此div4会自动上移，与div1组成一个“新”标准流，而浮动是漂浮在标准流之上，因此div2又挡住了div4。

咳咳，到重点了，当同时对div2、div3设置浮动之后，div3会跟随在div2之后，不知道读者有没有发现，一直到现在，div2在每个例子中都是浮动的，但并没有跟随到div1之后。因此，我们可以得出一个重要结论：

假如某个div元素A是浮动的，如果A元素上一个元素也是浮动的，那么A元素会跟随在上一个元素的后边(如果一行放不下这两个元素，那么A元素会被挤到下一行)；如果A元素上一个元素是标准流中的元素，那么A的相对垂直位置不会改变，也就是说A的顶部总是和上一个元素的底部对齐。

# 2 常考元素布局方法

## 2.1 常见的实现左右两栏布局的方法

·通过float浮动效果

·通过flex 实现左右两栏布局

·通过position absolute 绝对定位元素。

## 2.2 隐藏元素的方法有哪些

·**display: none**：渲染树不会包含该渲染对象，因此该元素不会在页面中占据位置，也不会响应绑定的监听事件

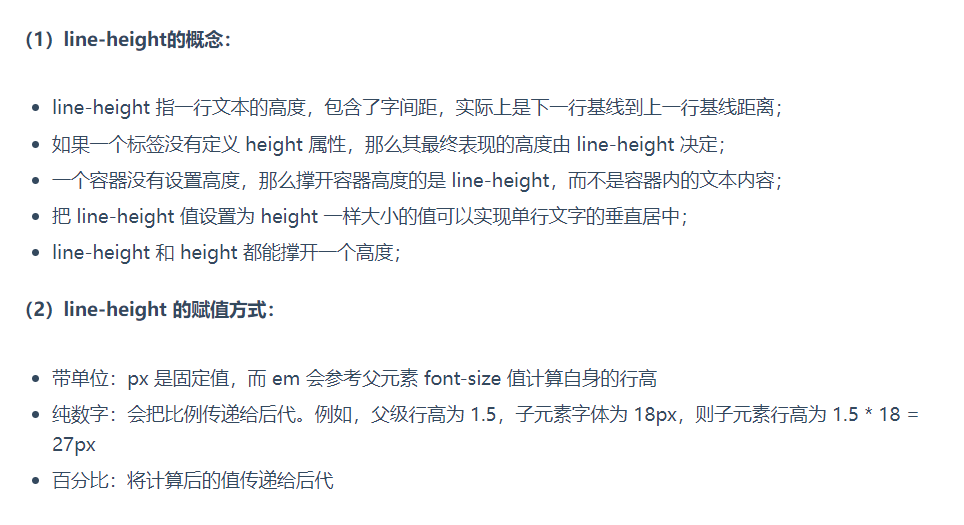
·**visibility: hidden**：元素在页面中仍占据空间，但是不会响应绑定的监听事件

·**opacity: 0**：将元素的透明度设置为 0，以此来实现元素的隐藏。元素在页面中仍然占据空间，并且能够响应元素绑定的监听事件

·**position: absolute**：通过使用绝对定位将元素移除可视区域内，以此来实现元素的隐藏

·**z-index: 负值**：来使其他元素遮盖住该元素，以此来实现隐藏

## 2.3 对line-height 的理解



## 2.4 判断元素是否到达可视区域

## 2.5 常见的CSS布局单位

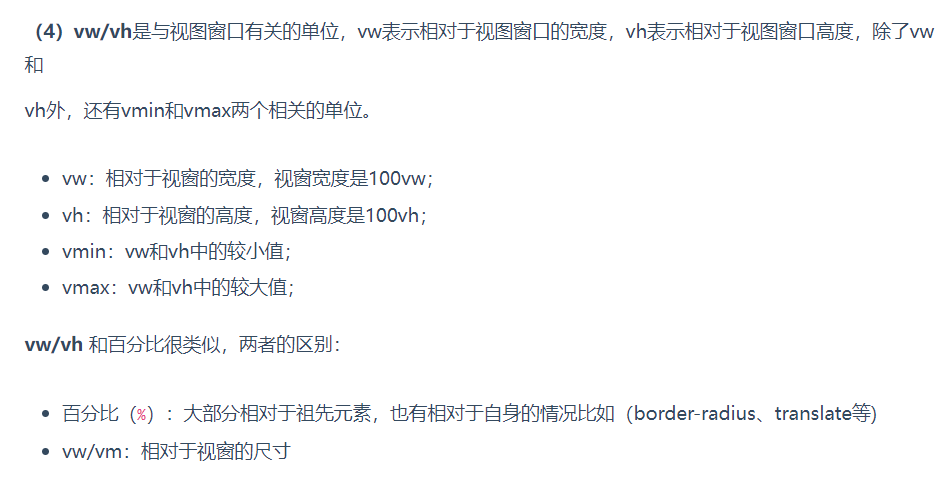
常用的布局单位包括像素（px），百分比（%），em，rem，vw/vh。

**像素**（px）是页面布局的基础，一个像素表示终端（电脑、手机、平板等）屏幕所能显示的最小的区域，像素分为两种类型：CSS像素和物理像素。

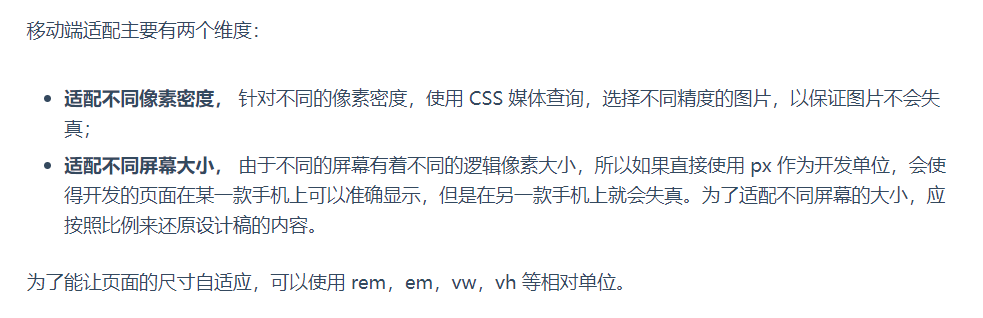
**百分比**（%），当浏览器的宽度或者高度发生变化时，通过百分比单位可以使得浏览器中的组件的宽和高随着浏览器的变化而变化，从而实现响应式的效果。一般认为子元素的百分比相对于直接父元素。

**em和rem**相对于px更具灵活性，它们都是相对长度单位，它们之间的区别：**em相对于父元素，rem相对于根元素。**

**em：** 文本相对长度单位。相对于当前对象内文本的字体尺寸。如果当前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(默认16px)。(相对父元素的字体大小倍数)



## 2.6 如何根据设计稿进行移动端适配



## 2.7 清除浮动

**浮动的定义：** 非IE浏览器下，容器不设高度且子元素浮动时，容器高度不能被内容撑开。 此时，内容会溢出到容器外面而影响布局。这种现象被称为浮动（溢出）

**浮动的工作原理**

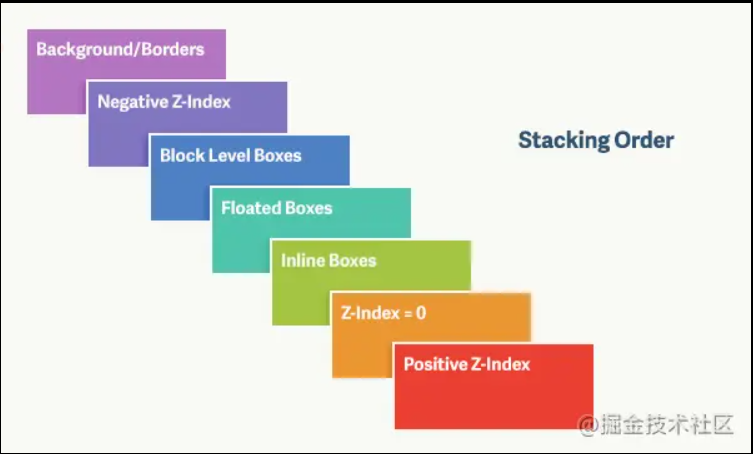
·浮动元素脱离文档流，不占据空间（引起“高度塌陷”现象）

·浮动元素碰到包含它的边框或者其他浮动元素的边框停留

浮动元素可以左右移动，直到遇到另一个浮动元素或者遇到它外边缘的包含框。浮动框不属于文档流中的普通流，当元素浮动之后，不会影响块级元素的布局，只会影响内联元素布局。此时文档流中的普通流就会表现得该浮动框不存在一样的布局模式。当包含框的高度小于浮动框的时候，此时就会出现“高度塌陷”。



## 2.8 元素的层叠顺序



上图，由上到下分别是：

（1）背景和边框：建立当前层叠上下文元素的背景和边框。

（2）负的z-index：当前层叠上下文中，z-index属性值为负的元素。

（3）块级盒：文档流内非行内级非定位后代元素。

（4）浮动盒：非定位浮动元素

（5）行内盒：文档流内行内级非定位后代元素。

（6）z-index:0：层叠级数为0的定位元素。

（7）正z-index：z-index属性值为正的定位元素

## 2.9 常见居中方案

·行内元素水平居中：直接使用text-align：center

·行内元素垂直居中：vertical-align：middle并设置父级元素的行高为父级元素的高度

·flex弹性布局：设置justify-content：center水平居中，align-items：垂直居中

·高度确定的时候：绝对定位+margin来垂直居中

·通过padding属性进行垂直居中

## 2.10 如何居中p 如何居中浮动元素 如何让绝对定位的p居中



## 2.11 常见的inline元素、block元素、inline-block元素

**常见的inline内联元素：**

* 和其他元素都在一行上，不会开始新的一行
* 元素的高度、宽度、行高及顶部和底部边距不可设置；
* 元素的宽度就是它包含的文字或图片的宽度，不可改变。

span、img、a、lable、input、abbr（缩写）、em（强调）、big、cite（引用）、i（斜体）、q（短引用）、textarea、select、small、sub、sup，strong、u（下划线）、button（默认display：inline-block））

**常见的block块级元素：**

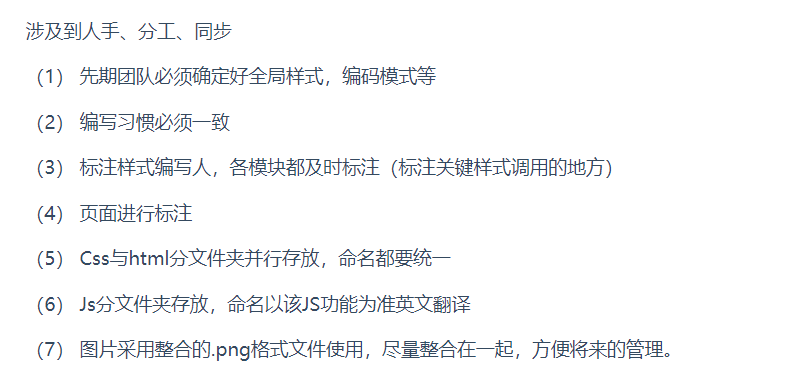
* 每个块级元素都从新的一行开始，并且其后的元素也另起一行。（一个块级元素独占一行）;
* 元素的高度、宽度、行高以及顶和底边距都可设置;
* 元素宽度在不设置的情况下，是它本身父容器的100%（和父元素的宽度一致），除非设定一个宽度。

div、p、h1…h6、ol、ul、dl、table、address、blockquote、form

**常见的inline-block内联块元素：**

img、input

## 2.12 如何来管理所有CSS文件,js 与图片



## 2.13 CSS auto



# 3 CSS基础知识

## 3.1css 的渲染层

在 DOM 树中每个节点都会对应一个渲染对象（RenderObject），当它们的渲染对象处于相同的坐标空间（z 轴空间）时，就会形成一个 RenderLayers，也就是渲染层。渲染层将保证页面元素以正确的顺序堆叠，这时候就会出现**层合成（composite）**，从而正确处理透明元素和重叠元素的显示。

## 3.2 CSS三大特性——优先级

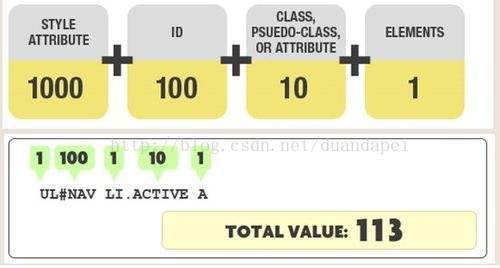
首先了解下CSS的三种样式类型：外部样式、内部样式以及内联样式。

内联样式  >  内部样式  >  外部样式

外部样式：外部样式与HTML代码分离，单独存放在.css文件中。使用时通过link标签将此文件引入到页面中，引入的格式为：<link type="text/css" rel="stylesheet" href=".css文件的地址"/>

内部样式：内部样式使用<style>标签将CSS代码包裹起来，并放入<head>标签中(一般存放在head中，也可存放于别处，如body中)，格式为：<style type="text/css"> span{color: red;} </style>

内联样式：内联样式定义在dom元素的style属性中，格式为：<div style="background-color: red;"> hello </div>



上图所表达的意思为：内联样式style属性的权值为1000，id选择器的权值为100，类、伪类、属性选择器的权值为10，标签选择器的权值为1。(权值越大，优先级越高)。

结合选择器的优先级，重新去看上面所提到的三种CSS样式类型的优先级，我们可以对这个优先级排序做如下解释：

1. 内联样式的优先级最高，是因为style标签的权值高。

2. 内部样式的优先级高于外部样式，是因为我们普遍将内部样式定义在外部样式之后。这样对于相同权重的样式，后面的会覆盖掉前面的(也可以理解为越靠近dom元素的样式，优先级相对越高)，所以最终呈现出来的是内部样式的优先级更高。我们下面做个测试，将内部样式定义在外部样式之前，看看效果如何：

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html" charset="utf-8"/>

<script src="Useful/jq.js"></script>

<style type="text/css">

#span1{

color: red;

}

</style>

<link type="text/css" rel="stylesheet" href="Useful/testPage.css"/>

<!-- testPage.css内容为：#span1{color: yellow;} -->

</head>

<body>

<span id="span1"> 我是span </span>

</body>

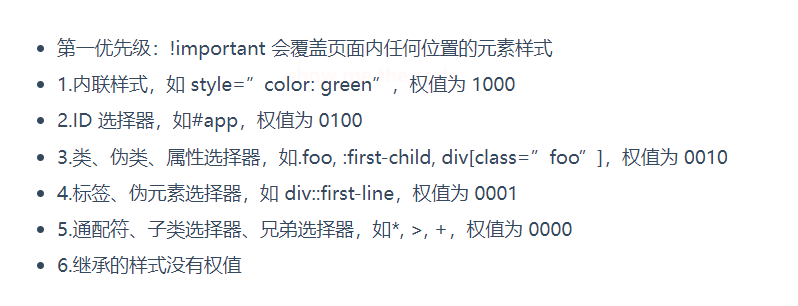
</html>

结果显示的是外部样式的颜色，没有显示为内部的红色

https://img-blog.csdn.net/20171101194316760?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvZHVhbmRhcGVp/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

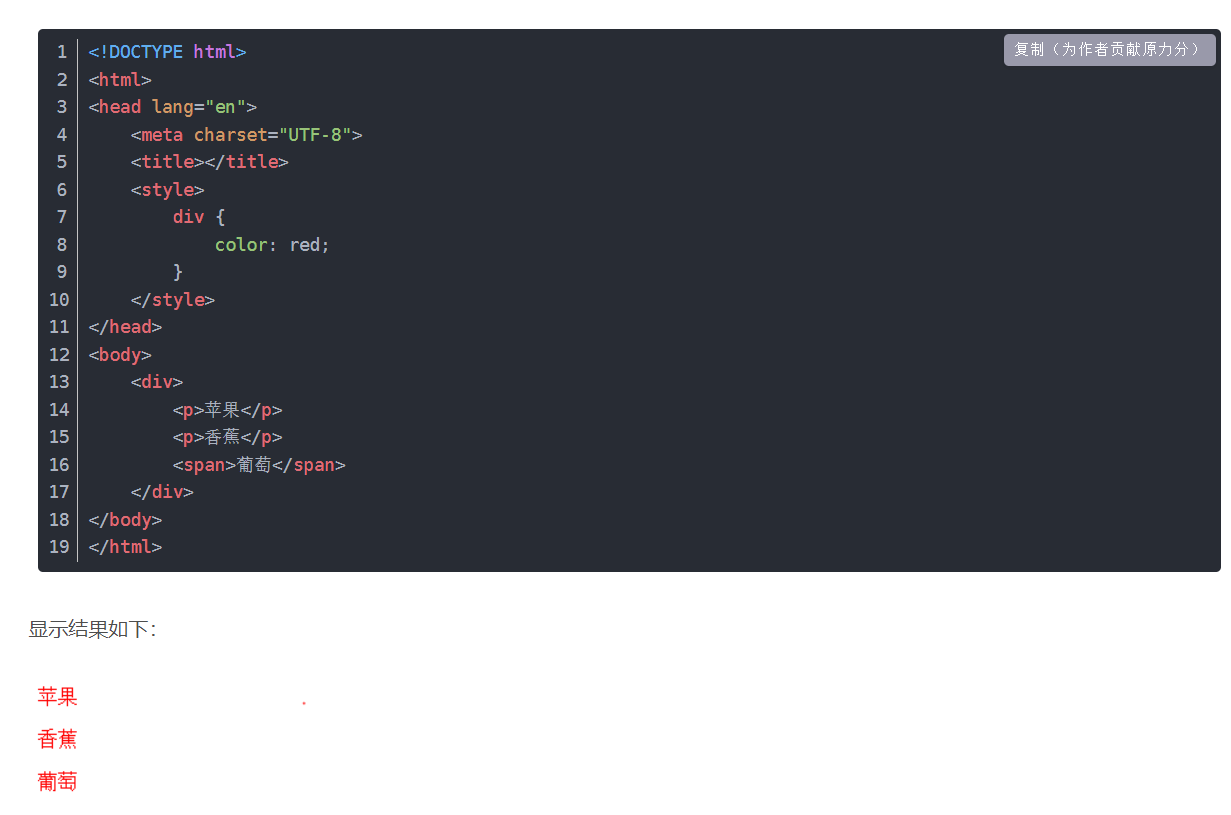
很显然，最终显示出来的是外部样式中定义的颜色。这就印证了内部样式和外部样式本身并没有优先级高低之分，最终的优先级是由选择器优先级和顺序所决定的。换一种方式理解：外部样式被link引入进页面后，所呈现出的形式类似于内部样式，他们是平等的。这时候决定最终样式的就是选择器和顺序了。

总结来说 优先级为：



## 3.3 CSS三大特性——继承级

继承性是指指被包在内部的标签将拥有外部标签的样式性，即子元素可以继承父元素的属性，例如下面的代码，div中包含2个p标签，1个span标签，当给div设置字体颜色为红色时，他的子标签会继承父元素的属性，因而会显示红色。 在CSS中以text-、font-、line- 开头的属性都是可以继承的。



## 3.4 CSS三大特性——层叠级

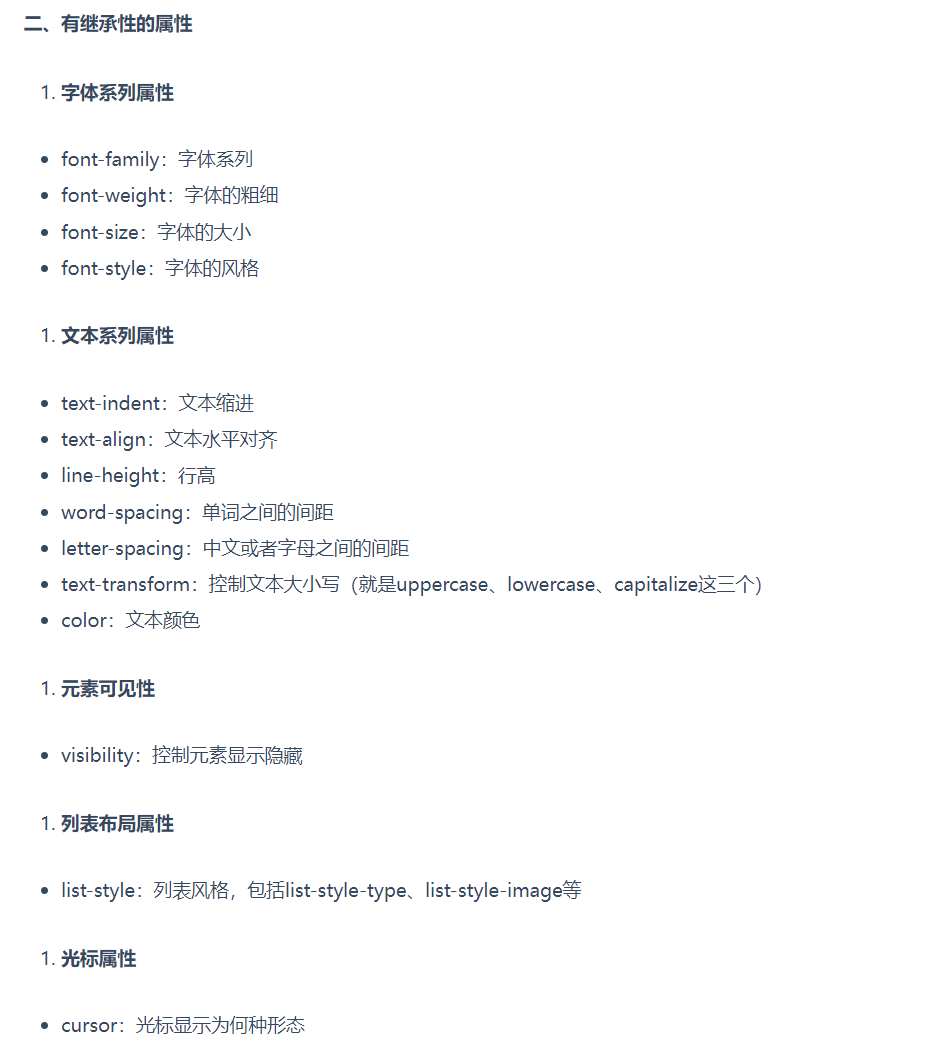
指多种CSS样式的叠加。  
是浏览器处理冲突的一个能力，如果一个属性通过两个相同选择器设置到同一个元素上，那么这个时候一个属性就会将另一个属性层叠掉。

原则：  
1、样式冲突，遵循的原则是就近原则，即CSS的书写位置。  
2、样式不冲突，不会层叠

z-index 属性设置元素的堆叠顺序。拥有更高堆叠顺序的元素总是会处于堆叠顺序较低的元素的前面。

## 3.5 CSS中可继承与不可继承属性有哪些

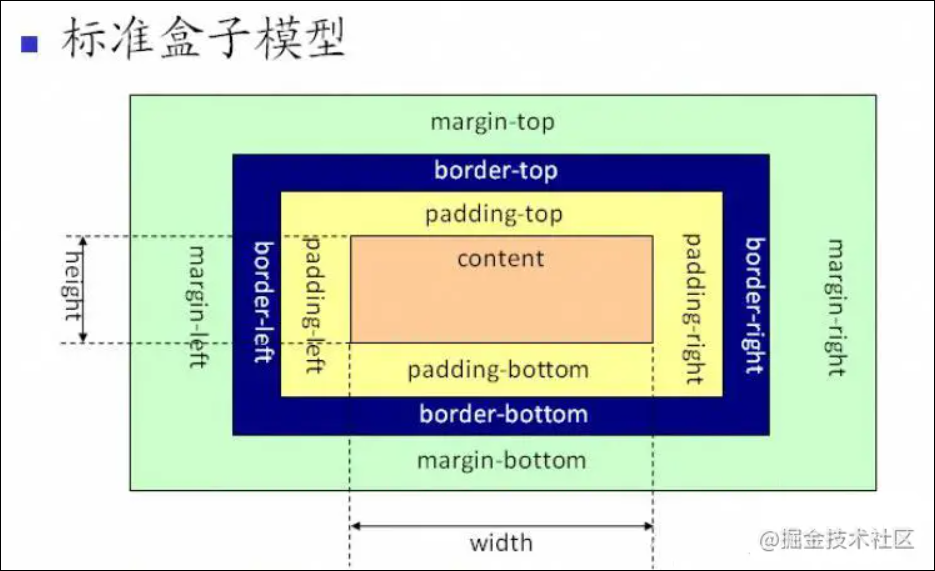


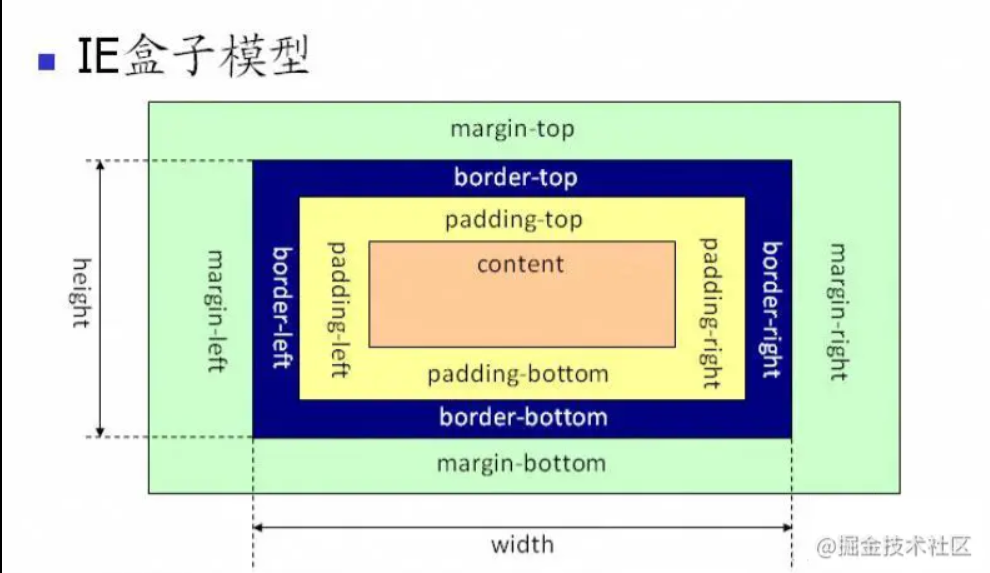


## 3.6 伪元素和伪类的区别和作用



## 3.7 CSS盒模型

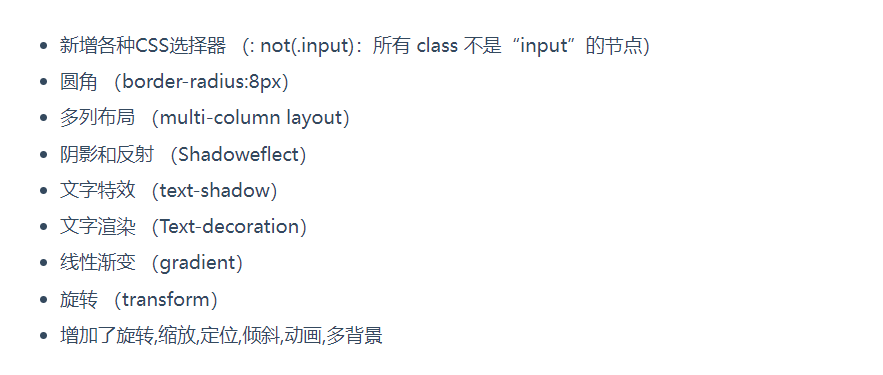




**标准盒模型的width和height属性的范围只包含了content**

**IE盒模型的width和height属性的范围包含了border、padding和content**

## 3.8 CSS3中有哪些新特性



## 3.9 CSS 优化和提高性能的方法有哪些

**加载性能优化**

·css压缩：将写好的css进行打包压缩，可以减小文件体积

**·**css单一样式：当需要下边距和左边距的时候，很多时候会选择使用 margin:top 0 bottom 0；但margin-bottom:bottom;margin-left:left;执行效率会更高

·减少使用@import，建议使用link，因为后者在页面加载时一起加载，前者是等待页面加载完成之后再进行

**选择器性能优化**

·关键选择器（key selector）。选择器的最后面的部分为关键选择器（即用来匹配目标元素的部分）。CSS选择符是从右到左进行匹配的。当使用后代选择器的时候，浏览器会遍历所有子元素来确定是否是指定的元素等等

·尽量少的去使用后代选择器，降低选择器的权重值。后代选择器的开销是最高的，尽量将选择器的深度降到最低，最高不要超过三层，更多的使用类来关联每一个标签元素。

·尽量少的去对标签进行选择，例如#app，而是用class

## 3.10 CSS预处理器/后处理器是什么

## 3.11 ::和 :的双冒号和单冒号有什么区别

**双冒号用于伪元素**

**单冒号用于伪类**

# 4 实战篇字节

## 4.1 字节CSS实战

### 4.1.1 用CSS实现一个三角形

将一个div的宽度和高度设置为0，然后设置边框样式

[复制代码](javascript:void(0);)

.triangle{

width: 0;

height: 0;

border-top: 50px solid black;

border-right: 50px solid red;

border-bottom: 50px solid green;

border-left: 50px solid blue;

}

[复制代码](javascript:void(0);)

将一个div的border都设置成50px粗，并且不同颜色，结果如下图所示：



把边框宽度设置成50px，计算机处理时，在边框交接处，一边占用一半的面积。

将左右下边框设置成transparent，就可以画出一个三角形

[复制代码](javascript:void(0);)

.triangle{

width: 0;

height: 0;

border-top: 50px solid black;

border-right: 50px solid transparent;

border-bottom: 50px solid transparent;

border-left: 50px solid transparent;

}

/\*下面代码更加简洁\*/

.triangle{

width: 0;

height: 0;

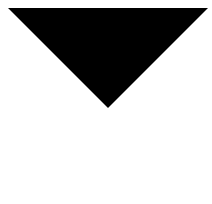
border: 50px solid transparent;

border-top-color: black ;

}

[复制代码](javascript:void(0);)

结果如下图所示：



将下边框的长度设置为0，实现一个最小空间的三角形

上面产生的三角形，实质上占位还是一个正方形，所以我们应该将这个三角形占用的空间尽可能缩小。不设置下边框

[复制代码](javascript:void(0);)

.triangle{

width: 0;

height: 0;

border-top: 50px solid black;

border-right: 50px solid transparent;

border-left: 50px solid transparent;

}

[复制代码](javascript:void(0);)

这样就可以实现高度为50px，宽度为100px的三角形，结果如下图所示：

