CSS类面试题目总结

## 用 CSS 实现元素垂直居中，有哪些好的方案？

链接：https://www.zhihu.com/question/20543196/answer/57757836

（1）不知道自己高度和父容器高度的情况下, 利用绝对定位只需要以下三行：

parentElement{

position:relative;

}

childElement{

position: absolute;

top: 50%;

transform: translateY(-50%);

}

（2）若父容器下只有一个元素，且父元素设置了高度，则只需要使用相对定位即可

parentElement{

height:xxx;

}

.childElement {

position: relative;

top: 50%;

transform: translateY(-50%);

}

（3）Flex 布局：

不考虑兼容老式浏览器的话，用Flex布局简单直观一劳永逸：

parentElement{

display:flex;/\*Flex布局\*/

display: -webkit-flex; /\* Safari \*/

align-items:center;/\*指定垂直居中\*/

}

## 水平垂直居中

## 块级元素、行内元素、内联块级元素

块级元素：结构性标记。**display:block;**

<div>、 <p>、<h1>、<form>、<ul> 、<li>。

特点：

1. 总是从新的一行开始
2. 高度、宽度、行高、顶和底边距都可设置
3. 宽度没有设置时，默认为100%
4. 块级元素中可以包含块级元素和行内元素

行内元素（内联元素）：描述性标记。**display:inline;**

<span>、<a>、<label>、 <strong> 和<em>

特点：

1. 和其他元素都在一行上；
2. 元素的高度、宽度及顶部和底部边距不可设置；
3. 元素的宽度就是它包含的文字或图片的宽度，不可改变。
4. 行内元素只能行内元素，包含块级元素语法不会报错，但是不提倡

内联块级元素：**display:inline-block;**

<img>、<input>

1. 和其他元素都在一行上
2. 元素的宽、高、行高、顶部和底部边距可设置

## CSS盒子模型

在一个文档中，每个元素都被表示为一个矩形的盒子。

盒子模型分为：W3C标准盒子模型、IE怪异盒子模型。模型描述了元素所占空间的内容。

所有HTML元素都可以看做盒子，包括：外边距，边框，填充，实际内容



注意！！！如果未设置box-sizing:border-box；设置的width和height是设置的内容区域的宽度和高度。

**W3C标准盒模型**：

元素的宽=content

盒子的宽：margin-left + border-left + padding-left + width + padding-right + border-right + margin-right

**IE怪异盒子模型：**

元素的宽=content + padding + border

盒子的宽= margin-left + width + margin-right

**解决方案：box-sizing**

// W3C 标准盒模型（浏览器默认）

box-sizing: content-box;

// IE 怪异盒模型

box-sizing: border-box;

当设置box-sizing:border-box;时，border和padding就被包含在了宽高之内，和IE盒模型是一样的。

为了避免CSS在不同浏览器上表现不同，写法最好为

\*, \*:before, \*:after {

-moz-box-sizing: border-box;

-webkit-box-sizing: border-box;

box-sizing: border-box;

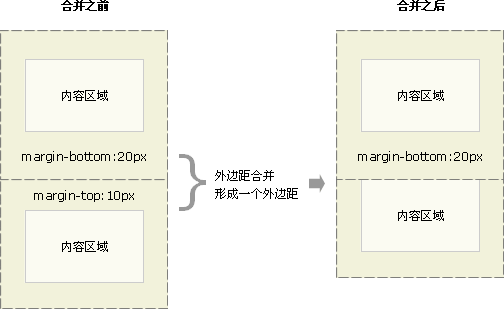
}

## 外边距合并（垂直）

外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。

合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

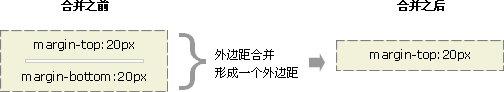
1. 一个元素出现在另一个元素上面，第一个元素的下边距与第二个元素的上边距合并。



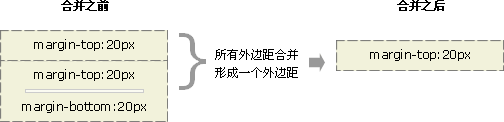
1. 一个元素包含在另一个元素中（外层元素没有设置内边距或边框把外边距分隔开）



1. 外边距与自身合并。一个空元素，有外边距，没有边框或填充，其上下外边距会合并



这个外边距遇到另一个元素的外边距，还会继续合并：



注意：只有普通文档流中块框的垂直外边距才会发生外边距合并。行内框、浮动框或绝对定位之间的外边距不会合并。

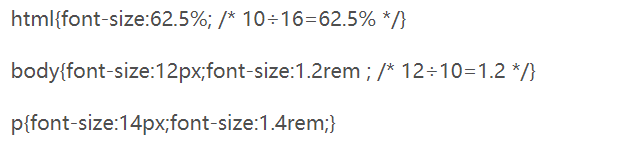
## CSS常用单位：px、em、rem

* 1. px：固定大小。一个像素等于电脑屏幕上的一个点（屏幕分辨率的最小分割）
  2. em：相对长度单位。相对于当前对象内文本的字体尺寸，如果当前对象内文本的字体尺寸未设置，则相对于浏览器的默认字体尺寸。
     1. em的值不是固定的
     2. em会继承父级元素的字体大小

浏览器的默认字体高是16px。所有未经调整的浏览器都符合：1em=16px。0.625em=10px。为了简化font-size的换算，在css中的body选择器中声明Font-size=62.5%，即将默认字体大小设置为16px\*62.5=10px。1em=10px。这样只需要将之前设置的px数值除以10，换上em作为单位即可。

缺点：由于em继承父元素的字体大小，当父元素字体改变时，需要重新计算。

* 1. rem：css3新增的相对长度单位，只相对根目录即HTML元素。只要在html标签上设置字体大小为标准，文档中的字体大小都会以此为参照



兼容性：IE9+，Firefox、Chrome、Safari、Opera 的主流版本都支持，为了兼容不支持 rem 的浏览器，我们需要在 rem 前面写上对应的 px 值，这样不支持的浏览器可以优雅降级。

* 1. %：类似于“em”。当使用百分比单位,你的文字在移动设备上仍然保持完全的可伸缩性和可访问性。1 em = 16 px = 100%

在一般的PC端网页制作过程中，px和em用的比较多，在移动端一般采用rem。

## 隐藏标签的方式

1. display：none；
2. visibility：hidden
3. opacity：0
4. 设置height，width等盒模型属性为0
5. 设置position，top，left等属性，将元素移出可视区域
6. clip-path

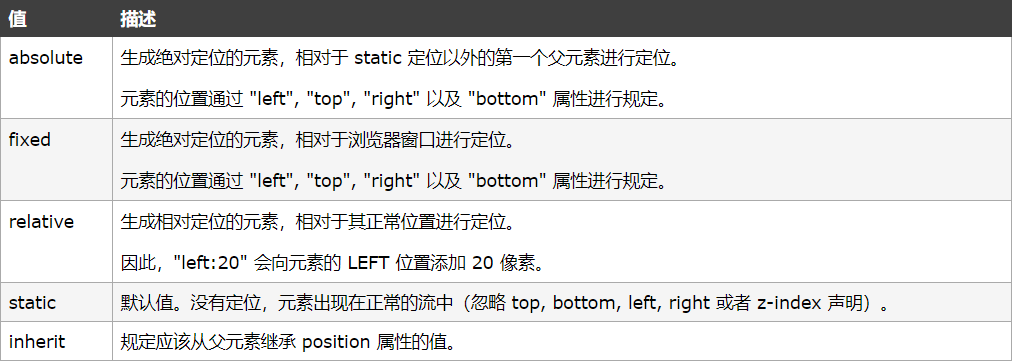
元素隐藏后的事件响应：

1. display:none：元素彻底消失，很显然不会触发其点击事件
2. visibility:hidden：无法触发其点击事件，有一种说法是display:none是元素看不见摸不着，而visibility:hidden是看不见摸得着，这种说法是不准确的，设置元素的visibility后无法触发点击事件，说明这种方法元素也是消失了，只是依然占据着页面空间。
3. opacity:0：可以触发点击事件，原因也很简单，设置元素透明度为0后，元素只是相对于人眼不存在而已，对浏览器来说，它还是存在的，所以可以触发点击事件
4. height:0：将元素的高度设置为0，并且设置overflow:hidden。使用这种方法来隐藏元素，是否可以触发事件要根据具体的情况来分析。如果元素设置了border，padding等属性不为0，很显然，页面上还是能看到这个元素的，触发元素的点击事件完全没有问题。如果全部属性都设置为0，很显然，这个元素相当于消失了，即无法触发点击事件。

通过JS可以触发被设置为display:none的元素的事件。无法触发点击事件的真正原因是鼠标无法真正接触到被设置成隐藏的元素！！！

## position属性

任何元素都可以定位，绝对或固定定位元素会生成一个块级框，无论元素是什么类型。相对定位元素会相对于他在正常流中的默认位置偏移。



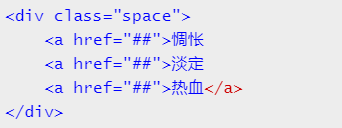
sticky：粘性定位。结合了relative和fixed定位。设置阈值（top | bottom | left | right），当未超过阈值时relative，到达阈值后fixed。

Margin-top和top的区别：

margin-top指的是自己的顶部距离容器其他元素的距离，top指的是自己的顶部距离容器顶部的距离，有点绝对定位的意思。所以margin-top通常与relative定位结合使用，top和absolute定位结合使用。

## 去除inline-block元素间间距的N种方法

1. 元素留白间距出现的原因：标签段之间的空格。
2. 去掉HTML中的空格，即连续书写标签，可读性差
3. margin负值
4. 不闭合标签。为了向下兼容IE6/IE7等喝蒙牛长大的浏览器，最后一个列表的标签的结束（闭合）标签不能丢。



1. 使用font-size:0。基本上可以解决大部分浏览器下inline-block元素之间的间距(IE7等浏览器有时候会有1像素的间距)。
2. 使用letter-spacing或word-spacing

## 轮播图实现原理

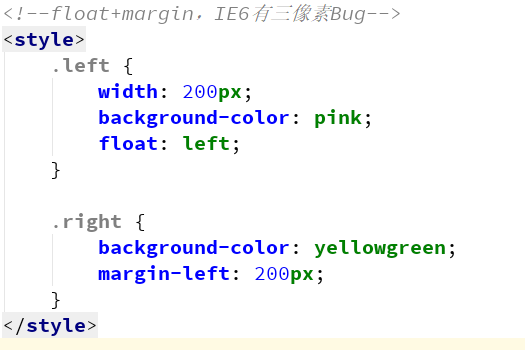
设置一个固定的父容器作为展示部分，overflow：hidden，一个子容器存放所有的轮播图片，通过设置子容器的position属性实现不同图片在父容器中的展示。例如左右切换的轮播，向左就设置子容器.left-=图片宽度，向右就设置子容器.left+=图片宽度。为了避免轮播的空白页面的出现，在第一张图片前增添最后一张图片，在最后一张图片后添加第一张图片。当到达边界值时，直接将left值设置为0或最大值；

## Bootstrap的使用，特性等

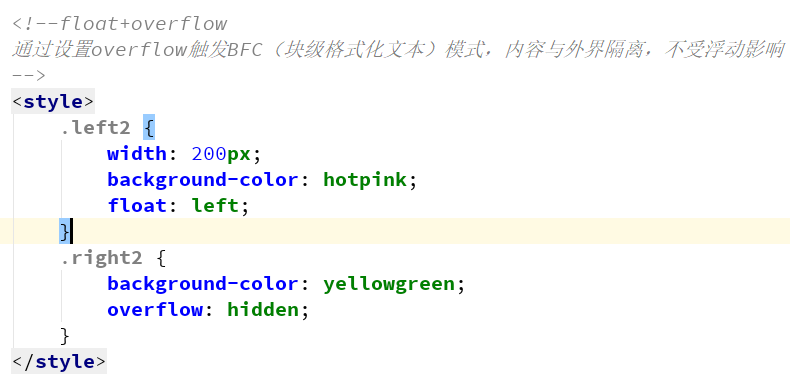
1. **less或sass，对CSS预编译器有没有了解**

## 两列布局，左定宽，右边宽度自适应

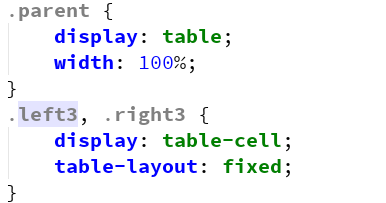
1. Float+margin：

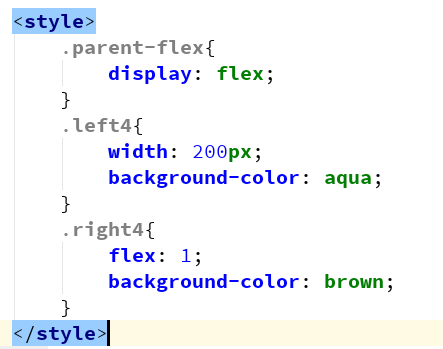


1. Float+overflow：hidden



1. Table



1. Flex
   * 1. 

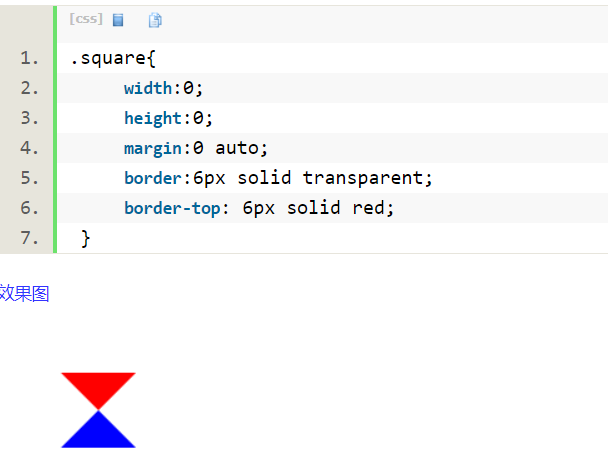
## 三角形实现方式

1. 使用border

第一步 保证元素是块级元素

第二步 设置元素的边框

第三步 不需要显示的边框使用透明色



1. 利用”◆“字符实现三角形：设置font-size和top/bottom/left/right值
2. 利用 CSS3 transfrom 旋转 45 度实现三角形：先进行top/bottom/left/right位移，再旋转

## 瀑布流布局

## CSS hack

针对不同的浏览器/不同版本写相应的CSS code的过程，叫做CSS hack!

### CSS hack的原理

由于不同的浏览器和浏览器各版本对CSS的支持及解析结果不一样，以及CSS优先级对浏览器展现效果的影响，我们可以据此针对不同的浏览器情景来应用不同的CSS。

### CSS hack分类

CSS Hack大致有3种表现形式，CSS属性前缀法、选择器前缀法以及IE条件注释法（即HTML头部引用if IE）Hack，实际项目中CSS Hack大部分是针对IE浏览器不同版本之间的表现差异而引入的。

* 属性前缀法(即类内部Hack)：例如 IE6能识别下划线"\_"和星号" \* "，IE7能识别星号" \* "，但不能识别下划线"\_"，IE6~IE10都认识"\9"，但firefox前述三个都不能认识。
* 选择器前缀法(即选择器Hack)：例如 IE6能识别\*html .class{}，IE7能识别\*+html .class{}或者\*:first-child+html .class{}。
* IE条件注释法(即HTML条件注释Hack)： 针对所有IE(注：IE10+已经不再支持条件注释)： <!--[if IE]>IE浏览器显示的内容 <![endif]-->，针对IE6及以下版本： <!--[if lt IE 6]>只在IE6-显示的内容 <![endif]-->。这类Hack不仅对CSS生效，对写在判断语句里面的所有代码都会生效。

### CSS hack利弊

一般情况下，我们尽量**避免使用**CSS hack，但是有些情况为了顾及用户体验实现向下兼容，不得已才使用hack。比如由于IE8及以下版本不支持CSS3,而我们的项目页面使用了大量 CSS3新属性在IE9/Firefox/Chrome下正常渲染，这种情况下如果不使用css3pie或htc或条件注释等方法时,可能就得让IE8- 的专属hack出马了。使用hack虽然对页面表现的一致性有好处，但过多的滥用会造成html文档混乱不堪，增加管理和维护的负担。

## @import和link之间的区别

**两者都是外部引用CSS的方式，但是存在一定的区别**：

* 老祖宗的差别。link是XHTML标签，除了加载CSS外，还可以定义RSS等其他事务；@import属于CSS范畴，只能加载CSS。
* 加载顺序的差别。link引用CSS时，在页面载入时同时加载；@import需要页面完全载入以后加载。
* 兼容性的差别。link是XHTML标签，无兼容问题；@import是在CSS2.1提出的，低版本的浏览器不支持。
* 使用dom控制样式时的差别。link支持使用Javascript控制DOM去改变样式；而@import不支持。

<**style**>  
 **@import url**(**"style.css"**);  
</**style**>

<**link rel="stylesheet" href="style.css"**>

## web页面重构怎么操作？

页面重构就是根据原有页面内容和结构的基础上，通过div+css写出符合web标准的页面结构。

具体实现要达到以下三点：

* 结构完整，可通过标准验证
* 标签语义化，结构合理
* 充分考虑到页面在站点中的“作用和重要性”，并对其进行有针对性的优化

## 清除浮动

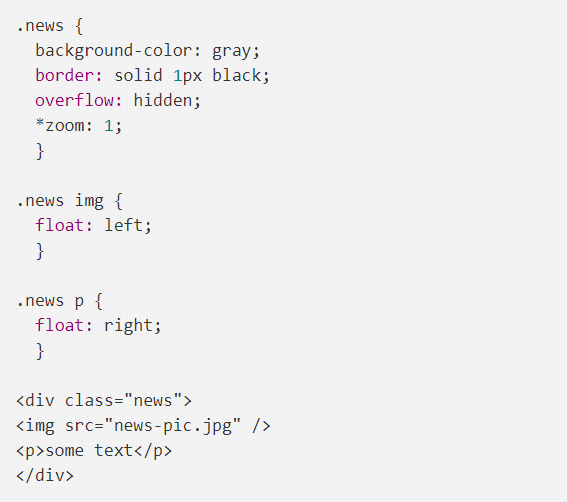
### ****1、使用空标签清除浮动****

空标签可以是div标签，也可以是P标签。也有很多人用<hr>，只是需要另外 为其清除边框，但理论上可以是任何标签。对于使用额外标签清除浮动（闭合浮动元素），是W3C推荐的 做法。

这种方式是在需要清除浮动的父级元素内部的所有浮动元素后添加这样一个标签清除浮动，并为其定义CSS代 码：clear:both。此方法的弊端在于增加了无意义的结构元素。

### 2、使用overflow属性。

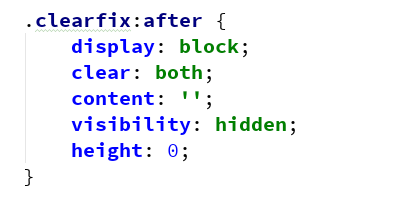
此方法有效地解决了通过空标签元素清除浮动而不得不增加无意代码的弊端：给浮动元素的容器添加overflow:auto或overflow：hidden。在IE6中需要触发hasLayout，为容器添加\*zoom:1； 本质上是使父容器形成BFC，BFC需要对浮动元素计算高度



### ****3、使用after伪对象清除浮动。****

 该方法只适用于非IE浏览器 。使用中需注意以下几点。一、该方法中必须为需要清除浮动元素的伪对象中设置height:0，否则该元素会比实际高出若干像 素；二、content属性是必须的，但其值可以为空。

在包裹浮动元素的父容器上设置：



清除浮动的方法分为两类：

* 1. clear属性，在浮动元素的末尾添加一个带有clear：both属性的空div
  2. 触发浮动元素父元素的BFC，使该父容器可以包含浮动元素

## 在网页中的应该使用奇数还是偶数的字体？

* 偶数字号相对更容易和 web 设计的其他部分构成比例关系。比如：当我用了 14 px 的正文字号，我可能会在一些地方用 14 × 0.5 = 7 px 的 margin，在另一些地方用 14 × 1.5 = 21 px 的标题字号。
* Windows 自带的点阵宋体（中易宋体）从 Vista 开始只提供 12、14、16 px 这三个大小的点阵，而 13、15、17 px 时用的是小一号的点阵（即每个字占的空间大了 1 px，但点阵没变），于是略显稀疏。

## box-sizing常用的属性有哪些？分别有什么作用？

(1) box-sizing: content-box|border-box|inherit;

　　(2) content-box:**宽高不包括边框和内边距，只包含内容区域。**宽度和高度分别应用到元素的内容框。在宽度和高度之外绘制元素的内边距和边框(元素默认效果)。

border-box:**宽高包括边框和内边距**。元素指定的任何内边距和边框都将在已设定的宽度和高度内进行绘制。通过从已设定的宽度和高度分别减去边框和内边距才能得到内容的宽度和高度。

## CSS优先级

4个等级的定义如下：

第一等级：代表内联样式，如style=""，权值为 1000

第二等级：代表id选择器，如#content，权值为100

第三等级：代表类，伪类和属性选择器，如.content，权值为10

第四等级：代表标签选择器和伪元素选择器，如div p，权值为1

注意：通用选择器（\*），子选择器（>），和相邻同胞选择器（+）并不在这个等级中，所以他们的权值为0

**内联》ID选择器》伪类=属性选择器=类选择器》元素选择器【p】》通用选择器(\*)》继承的样式**

## 块级格式化上下文BFC

### **Box**

Box是CSS布局的基本单位。直观点说，一个页面由多个Box组成，元素类型和display属性决定了Box的类型。不同类型的Box，会参与不同的Formatting Context（一个决定如何渲染文档的容器）。

盒子类型：

* + 1. block-level box：display属性为block，list-item，table的元素，会生成block-level box，并且参与block formatting context
    2. inline-level box：display属性为inline，inline-block，inline-table的元素，会生成inline-level box，并参与inline formatting context
    3. run-in box：CSS3中出现

### Formatting context

Formatting context 是 W3C CSS2.1 规范中的一个概念。它是页面中的一块渲染区域，并且有一套渲染规则，它决定了其子元素将如何定位，以及和其他元素的关系和相互作用。最常见的 Formatting context 有 Block fomatting context (简称BFC)和 Inline formatting context (简称IFC)。

CSS2.1 中只有 BFC 和 IFC, CSS3 中还增加了 GFC 和 FFC。

### BFC

1、BFC概念：块级格式化上下文。它是一个独立的渲染区域，只有block-level box参与，规定了内部的block-level box如何布局，与这个区域外部毫不相干。

1. 如何触发BFC
   1. 根元素body
   2. float不为none
   3. Position：absolute或fixed
   4. display：inline-block、table-cell、table-caption、flex、inline-flex
   5. overflow：hidden、auto、scroll。不是visible
2. BFC布局规则：
   1. BFC内部的box在垂直方向一个接一个的放置
   2. box垂直方向的距离由margin决定，属于同一个BFC的两个相邻Box的margin可能会发生叠加
   3. 每个元素的margin box的左边， 与包含块border box的左边相接触(对于从左往右的格式化，否则相反)。即使存在浮动也是如此
   4. BFC的区域不会与float box重叠
   5. BFC是一个隔离的独立容器，容器中的不影响外面，反之亦然
   6. 计算BFC高度时，浮动元素也参与计算
3. BFC的应用
   1. 自适应两栏布局：利用规则“BFC区域不会与float box重叠”
   2. 清除内部浮动：利用规则“浮动元素需要参与BFC高度计算”
   3. 防止margin重叠：将会发生重叠的元素放在不同的BFC中。规则“Box垂直方向的距离由margin决定。属于同一个BFC的两个相邻Box的margin会发生重叠”
4. 总结

所有都是围绕规则“**BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。反之也如此。**”

因为BFC内部的元素和外部的元素绝对不会互相影响，因此， 当BFC外部存在浮动时，它不应该影响BFC内部Box的布局，BFC会通过变窄，而不与浮动有重叠。同样的，当BFC内部有浮动时，为了不影响外部元素的布局，BFC计算高度时会包括浮动的高度。避免margin重叠也是这样的一个道理。

## CSS雪碧图

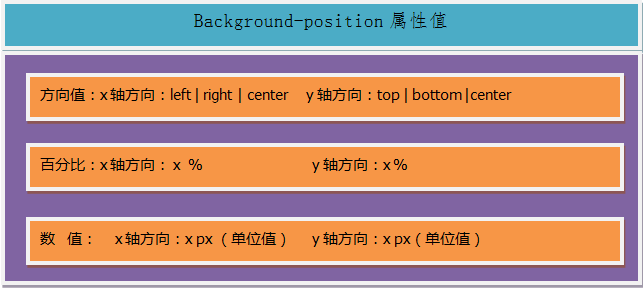
CSS雪碧图即CSS Spirit，又叫做CSS精灵，是一种CSS图像合并技术，将小图标和背景图合并到一张图上，然后利用CSS的背景定位来显示需要显示的图片部分。

使用background-position属性实现

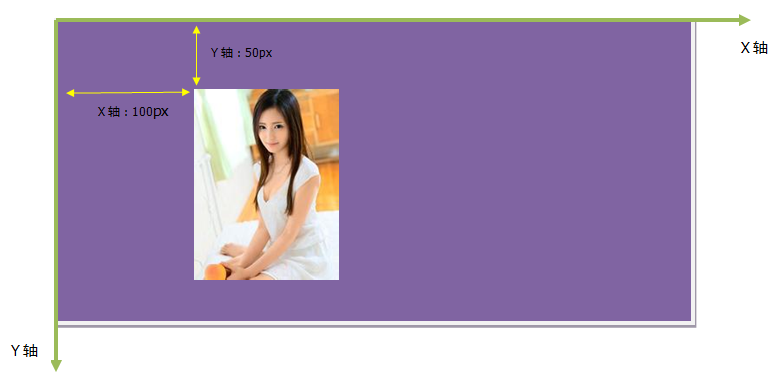
#### background-position

设置背景图像的起始位置，如果背景图要重复，从此位置开始。

有两个属性值： background-position：x | y，用法上可以对其一个属性单独使用 background-position-x 和 background-position-y。



* 1. 属性值为像素



* 1. 属性值为%

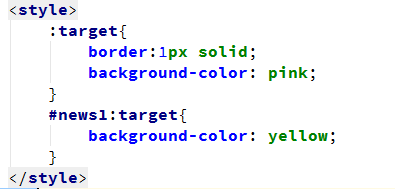
（容器自身的宽度/高度 - 图片自身的宽度/高度） x 百分比

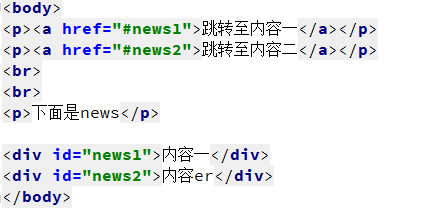


* 1. 不设置background-position：默认是0% 0%
  2. 方向值和百分比的计算方式是一样的，它们可以相互转换，left：0%，right：100%，center：50%。
  3. 如果background-position属性值只设置一个，那么另一个默认为center。

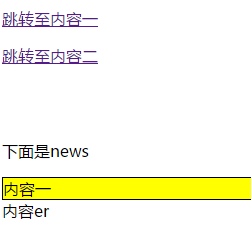
## CSS实现标签页动态切换效果

<a>标签的href属性锚点+：target伪类





点击“跳转至内容一”后：



## 文字水平垂直居中text-align、line-height、vertical-align的用法

参考：<http://www.jb51.net/css/29328.html>

text-align

line-height

base-line

## transition、transform、animation

1. transition：平滑的改变CSS的值。无论是点击事件，焦点事件，还是鼠标hover，只要值改变了，就是平滑的，就是动画。转换过程

transiton属性是下面几个属性的缩写：

* + 1. transition-property：指定过渡的属性值，比如transition-property:opacity就是只指定opacity属性参与这个过渡。
    2. transition-duration：指定这个过渡的持续时间
    3. transition-delay：延迟过渡时间
    4. transition-timing-function：指定过渡动画缓动类型，有ease | linear | ease-in | ease-out | ease-in-out | cubic-bezier()

其中，linear线性过度，ease-in由慢到快，ease-out由快到慢，ease-in-out由慢到快在到慢。

1. transform：拉伸（skew），压缩（scale），旋转（rotate），偏移（translate）。

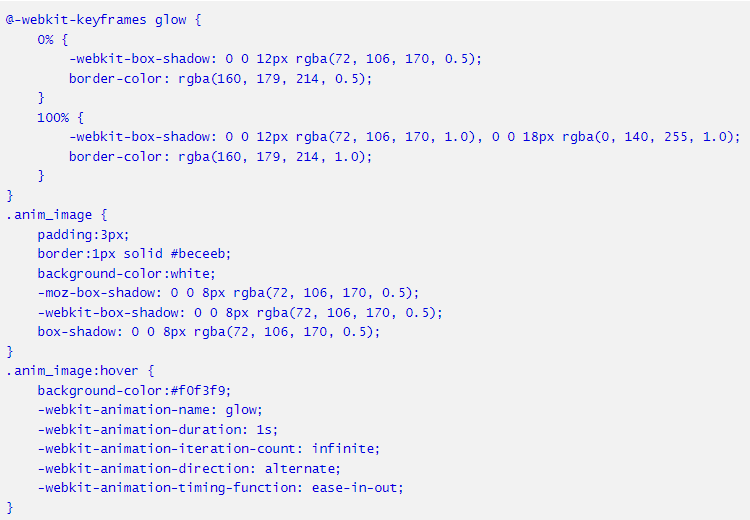
transform通常用来表示达到的效果

1. animation

与keyframes结合使用

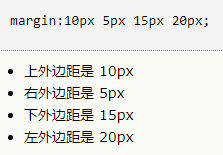
animation: name duration timing-function delay iteration-count direction;

|  |  |
| --- | --- |
| **值** | **描述** |
| [*animation-name*](http://www.w3school.com.cn/cssref/pr_animation-name.asp) | 规定需要绑定到选择器的 keyframe 名称。。 |
| [*animation-duration*](http://www.w3school.com.cn/cssref/pr_animation-duration.asp) | 规定完成动画所花费的时间，以秒或毫秒计。 |
| [*animation-timing-function*](http://www.w3school.com.cn/cssref/pr_animation-timing-function.asp) | 规定动画的速度曲线。 |
| [*animation-delay*](http://www.w3school.com.cn/cssref/pr_animation-delay.asp) | 规定在动画开始之前的延迟。 |
| [*animation-iteration-count*](http://www.w3school.com.cn/cssref/pr_animation-iteration-count.asp) | 规定动画应该播放的次数。 |
| [*animation-direction*](http://www.w3school.com.cn/cssref/pr_animation-direction.asp) | 规定是否应该轮流反向播放动画。 |



## margin、padding

margin：元素外边距。简写属性。顺时针方向设置。只有父容器有border或者padding时，margin-top属性才会正常，否则是只能是相对于body的margin-top的效果。



|  |  |
| --- | --- |
| **值** | **描述** |
| auto | 浏览器计算外边距。 |
| *length* | 规定以具体单位计的外边距值，比如像素、厘米等。默认值是 0px。 |
| *%* | 规定基于父元素的宽度的百分比的外边距。 |
| inherit | 规定应该从父元素继承外边距。 |

padding：内填充。取值与margin相同。但是，margin允许负值，padding不允许

## CSS 布局模型

布局模型是建立在盒模型基础之上的。在网页中，元素有三种布局模型：流动模型（Flow）、浮动模型(Float)、层模型(Layer)。

### 流动模型

流动模型是默认的网页布局模式。

具有两个典型特征：

* + - 1. 块级元素都会在所处的包含元素内自上而下按顺序垂直延伸分布，因为在默认状态下，块级元素的宽度都为100%。
      2. 内联元素都会在所处元素内从左到右水平分布显示

### 浮动模型

float：left; float:right;

### 层模型

层模型就像是PS中的图层编辑功能一样，每个图层能够精确定位操作。

定位：允许你定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置。

层模型有三种模式：

* + 1. 绝对定位（position：absolute）：将元素从文档流中拖出来，然后使用left、right、top、bottom属性相对于其最接近的一个**具有定位属性的父包含块**进行绝对定位。如果不存在这样的包含块，则相对于body元素，即相对于**浏览器窗口**。
    2. 相对定位（position：relative）：通过left、right、top、bottom属性确定元素在**正常文档流**中的偏移位置。相对定位首先按static（float）方式生成一个元素（并且元素像层一样浮动了起来），然后相对于**以前的位置**移动，移动方向和幅度由设置的left、right、top、bottom属性确定，偏移前的位置保留不动。
    3. 固定定位（position：fixed）：与absolute定位相似，相对移动的坐标是视图（**屏幕内网页窗口**）本身。由于视图本身是固定的，它不会随浏览器窗口的滚动条滚动而变化，除非你在屏幕中移动浏览器窗口的屏幕位置，或改变浏览器窗口的显示大小，因此固定定位的元素会始终位于浏览器窗口内视图的某个位置，**未找到索引项。**不会受文档流动影响，这与background-attachment:fixed;属性功能相同。

## 浏览器的回流与重绘

### 浏览器渲染规则

1. 浏览器使用流式布局模型 (Flow Based Layout)。
2. 浏览器会把HTML解析成DOM，把CSS解析成CSSOM，DOM和CSSOM合并就产生了Render Tree。
3. 有了RenderTree，我们就知道了所有节点的样式，然后计算他们在页面上的大小和位置，最后把节点绘制到页面上。
4. 由于浏览器使用流式布局，对Render Tree的计算通常只需要遍历一次就可以完成，但table及其内部元素除外，他们可能需要多次计算，通常要花3倍于同等元素的时间，这也是为什么要避免使用table布局的原因之一。

### 回流（Reflow）

当Render Tree中部分或全部元素的**尺寸、结构、或某些属性**发生改变时，浏览器重新渲染**部分或全部文档**的过程称为回流。

会导致回流的操作：

* 页面首次渲染
* 浏览器窗口大小发生改变
* 元素尺寸或位置发生改变
* 元素内容变化（文字数量或图片大小等等）
* 元素字体大小变化
* 添加或者删除**可见**的DOM元素
* 激活CSS伪类（例如：:hover）
* 查询某些属性或调用某些方法

一些常用且会导致回流的属性和方法：

* clientWidth、clientHeight、clientTop、clientLeft
* offsetWidth、offsetHeight、offsetTop、offsetLeft
* scrollWidth、scrollHeight、scrollTop、scrollLeft
* scrollIntoView()、scrollIntoViewIfNeeded()
* getComputedStyle()
* getBoundingClientRect()
* scrollTo()

上述这些操作会导致引擎重新渲染来确保获取的值是实时的

### 重绘（Repaint）

当页面中元素样式的改变并不影响它在文档流中的位置时（例如： color、background-color、visibility等），浏览器会将新样式赋予给元素并重新绘制它，这个过程称为重绘。

### 性能影响

回流比重绘的代价更高。回流一定会导致重绘。

现代浏览器会对频繁的回流或重绘操作进行优化：

浏览器会维护一个队列，把所有引起回流和重绘的操作放入队列中，如果队列中的任务数量或者时间间隔达到一个阈值的，浏览器就会将队列清空，进行一次批处理，这样可以把多次回流和重绘变成一次。

当你访问以下属性或方法时，浏览器会立刻清空队列：

* clientWidth、clientHeight、clientTop、clientLeft
* offsetWidth、offsetHeight、offsetTop、offsetLeft
* scrollWidth、scrollHeight、scrollTop、scrollLeft
* width、height
* getComputedStyle()
* getBoundingClientRect()

因为队列中可能会有影响到这些属性或方法返回值的操作，即使你希望获取的信息与队列中操作引发的改变无关，浏览器也会强行清空队列，确保你拿到的值是最精确的。

### 如何避免

**CSS**

* 避免使用table布局。
* 尽可能在DOM树的最末端改变class。
* 避免设置多层内联样式。
* 将动画效果应用到position属性为absolute或fixed的元素上。
* 避免使用CSS表达式（例如：calc()）。

**JavaScript**

* 避免频繁操作样式，最好一次性重写style属性，或者将样式列表定义为class并一次性更改class属性。
* 避免频繁操作DOM，创建一个documentFragment，在它上面应用所有DOM操作，最后再把它添加到文档中。
* 也可以先为元素设置display: none，操作结束后再把它显示出来。因为在display属性为none的元素上进行的DOM操作不会引发回流和重绘。
* 避免频繁读取会引发回流/重绘的属性，如果确实需要多次使用，就用一个变量缓存起来。
* 对具有复杂动画的元素使用绝对定位，使它脱离文档流，否则会引起父元素及后续元素频繁回流。

## 动画

### transition

**平滑的改变CSS值，定义css属性的变化过程。 一般由**点击事件、焦点事件、hover等触发。属性如下所示：

* transition-property 过渡的性质，比如background过渡。
* transition-duration 过渡持续的时间，比如2s
* transition-delay 延迟过渡时间，比如1s
* transition-timing-function 指定过渡类型，有ease|linear|ease-in|ease-out|ease-in-out|cubic-bezier

一般将属性全部用transition写在一起，并且**由于兼容性的问题，**需要添加各个浏览器前缀。

### transform

变换，定义css样式的变化结果。即拉伸、压缩、旋转、偏移等

.trans\_skew { transform: skew(35deg); }

.trans\_scale { transform:scale(1, 0.5); }

.trans\_rotate { transform:rotate(45deg); }

.trans\_translate { transform:translate(10px, 20px); }

可结合transition实现动画效果

### animation

为了代替flash。

使用@keyframes规则创建动画，@keyframes规则定义了一个css样式和动画将逐步从目前的样式更新为新的样式。

animation: 动画名称（@keyframes规定的名称） 动画时长

### 三者比较

Transition：强调过渡；需要触发一个事件，例如鼠标点击，hover、焦点

Animation：多个关键帧，实现自由动画，不需要触发事件，通过@keyframes控制当前帧属性更灵活

Transform：静态变换结果，需要和transition或animation结合才能实现动画效果