

CSS

css实现一个对话气泡，有尖角的提示框

```
<div id="demo"></div>
<style>
  #demo {
    width: 100px;
    height: 80px;
    background-color: #ccc;
    position: relative;
    border: 4px solid #333;
  }
  #demo:after, #demo:before {
    border: solid transparent;
    content: ' ';
    height: 0;
    left: 100%;
    position: absolute;
    width: 0;
  }
  #demo:after {
    border-width: 9px;
    border-left-color: #ccc;
    top: 15px;
  }
  #demo:before {
    border-width: 14px;
    border-left-color: #333;
    top: 10px;
  }
</style>
```

使用css实现一个持续的动画效果

```

animation: mymove 5s infinite;
@keyframes mymove {
  from {top:0px;}
  to {top:200px;}
}

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

```

CSS清除浮动大全共8种方法（父级div定义 height、结尾处加空div标签 clear:both、父级div定义 伪类:after 和 zoom、父级div定义 overflow:hidden、父级div定义 overflow:auto、父级div 也一起浮动、父级div定义 display:table、结尾处加 br标签 clear:both）

1、父级div手动定义height，就解决了父级div无法自动获取到高度的问题

```

<style>
  .div1{background:#000080;border:1px solid red; /*解决代码*/height:200px;}
  .div2{background:#800080;border:1px solid red;height:100px;margin-top:10px}
  .left{float:left;width:20%;height:200px;background:#DDD}
  .right{float:right;width:30%;height:80px;background:#DDD}
</style>
<div class="div1">
  <div class="left">Left</div>
  <div class="right">Right</div>
</div>
<div class="div2">div2</div>

```

优点：简单、代码少、容易掌握

缺点：只适合高度固定的布局，要给出精确的高度，如果高度和父级div不一样时，会产生问题

建议：不推荐使用，只建议高度固定的布局时使用

2、结尾处加空div标签 clear:both

```

<style type="text/css">
    .div1{background:#000080;border:1px solid red}
    .div2{background:#800080;border:1px solid red;height:100px;margin-top:10px}
    .left{float:left;width:20%;height:200px;background:#DDD}
    .right{float:right;width:30%;height:80px;background:#DDD}
    /*清除浮动代码*/
    .clearfloat{clear:both}
</style>
<div class="div1">
    <div class="left">Left</div>
    <div class="right">Right</div>
    <div class="clearfloat"></div>
</div>
<div class="div2">div2</div>

```

原理：添加一个空div，利用css提高的clear:both清除浮动，让父级div能自动获取到高度

优点：简单、代码少、浏览器支持好、不容易出现怪问题

缺点：不少初学者不理解原理；如果页面浮动布局多，就要增加很多空div，让人感觉很不好

建议：不推荐使用，但此方法是以前主要使用的一种清除浮动方法

3、父级div定义 伪类:after 和 zoom

```

<style type="text/css">
    .div1{background:#000080;border:1px solid red;}
    .div2{background:#800080;border:1px solid red;height:100px;margin-top:10px}
    .left{float:left;width:20%;height:200px;background:#DDD}
    .right{float:right;width:30%;height:80px;background:#DDD}
    /*清除浮动代码*/
    .clearfloat:after{display:block;clear:both;content:"";visibility:hidden;height:0}
    .clearfloat{zoom:1}
</style>
<div class="div1 clearfloat">
    <div class="left">Left</div>
    <div class="right">Right</div>
</div>
<div class="div2">div2</div>

```

原理：IE8以上和非IE浏览器才支持:after，原理和方法2有点类似，zoom(IE特有属性)可解决ie6,ie7浮动问题

优点：浏览器支持好、不容易出现怪问题（目前：大型网站都有使用，如：腾讯，网易，新浪等等）

缺点：代码多、不少初学者不理解原理，要两句代码结合使用才能让主流浏览器都支持。

建议：推荐使用，建议定义公共类，以减少CSS代码。

4、父级div定义 overflow:hidden

```

<style type="text/css">
    .div1{background:#000080;border:1px solid red; /*解决代码*/width:98%;overflow:hidden}
    .div2{background:#800080;border:1px solid red;height:100px;margin-top:10px;width:98%}
    .left{float:left;width:20%;height:200px;background:#DDD}
    .right{float:right;width:30%;height:80px;background:#DDD}
</style>
<div class="div1">
    <div class="left">Left</div>
    <div class="right">Right</div>
</div>
<div class="div2">div2</div>

```

原理：必须定义width或zoom:1，同时不能定义height，使用overflow:hidden时，浏览器会自动检查浮动区域的高度

优点：简单、代码少、浏览器支持好

缺点：不能和position配合使用，因为超出的尺寸的会被隐藏。

建议：只推荐没有使用position或对overflow:hidden理解比较深的朋友使用。

5、父级div定义 overflow:auto

```

<style type="text/css">
    .div1{background:#000080;border:1px solid red; /*解决代码*/width:98%;overflow:auto}
    .div2{background:#800080;border:1px solid red;height:100px;margin-top:10px;width:98%}
    .left{float:left;width:20%;height:200px;background:#DDD}
    .right{float:right;width:30%;height:80px;background:#DDD}
</style>
<div class="div1">
    <div class="left">Left</div>
    <div class="right">Right</div>
</div>
<div class="div2">div2</div>

```

原理：必须定义width或zoom:1，同时不能定义height，使用overflow:auto时，浏览器会自动检查浮动区域的高度

优点：简单、代码少、浏览器支持好

缺点：内部宽高超过父级div时，会出现滚动条。

建议：不推荐使用，如果你需要出现滚动条或者确保你的代码不会出现滚动条就使用吧。

1、请用DIV在宽度为百分比的情况下绘制一个正方形；

```

<div style="width:30%;padding-bottom:30%;height:0px;background:red;"></div>

```

2、css实现会话上面的三角尖，用css实现一个三角形

```

.box{
    width: 0;
    height: 0;
    border-width: 40px;
    border-style: solid;
    border-color: red transparent transparent transparent;
}

```

3、css实现一个黑白相间的背景

```
.box{
  width:100%;
  height:400px;
  background-image: linear-gradient(to right,#000 50%, #fff 10%, #000 50%, #fff 10%);
  background-size:50px 100%;
}
```

4、position的relative和position定义，区别体现；外层是position呢，里面还是position呢

Position定位：

static:HTML元素的默认值，即没有定位，元素出现在正常的流中。静态定位的元素不会受到 top, bottom, left, right影响。

fixed:元素的位置相对于浏览器窗口是固定位置。Fixed定位使元素的位置与文档流无关，因此不占据空间。

Relative:

- 1、相对定位元素的定位是相对其正常位置。
- 2、可以移动的相对定位元素的内容和相互重叠的元素，它原本所占的空间不会改变。
- 3、相对定位元素经常被用来作为绝对定位元素的容器块。
- 4、relative-relative relative-absolute

absolute:

- 1、绝对定位的元素的位置相对于最近的已定位父元素，如果元素没有已定位的父元素，那么它的位置相对于<html>:
- 2、absolute 定位使元素的位置与文档流无关，因此不占据空间。
- 3、absolute 定位的元素和其他元素重叠。
- 4、relative-absolute absolute-absolute

5、盒模型,flex盒模型区别？

box-flex是旧的规则，flex是新的，弹性盒模型的两种容器块级伸缩容器和内联伸缩容器的区别类似于block和inline-block的区别，一个独占一行，另一个非独占一行

6. display为none和visibility为hidden区别

- 1.display:none是彻底消失，不在文档流中占位，浏览器也不会解析该元素；visibility:hidden是视觉上消失了，可以理解为透明度为0的效果，在文档流中占位，浏览器会解析该元素；
- 2.使用visibility:hidden比display:none性能上要好，display:none切换显示时visibility，页面产生回流（当页面中的一部分元素需要改变规模尺寸、布局、显示隐藏等，页面重新构建，此时就是回流。所有页面第一次加载时需要产生一次回流），而visibility切换是否显示时则不会引起回流。

7、让DIV垂直居中多种方法：

1、

```
body{text-align:center;vertical-align:middle}
```

2、div绝对定位水平垂直居中，

```
.div{ margin:0 auto; }
```

3、div绝对定位水平垂直居中【margin 负间距】

```
.div{ position: absolute;left:50%;top:50%;margin-left:-width/2;margin-top:-height/2; }
```

4、div绝对定位水平垂直居中【Transforms 变形】

```
.div{
    width: 200px;
    height: 200px;
    background: green;
    position: absolute;
    left: 50%;    /* 定位父级的50% */
    top: 50%;
    transform: translate(-50%, -50%); /* 自己的50% */
}
```

5、css不定宽高水平垂直居中

```
.box{
    height: 600px;
    display: flex;
    justify-content: center;
    align-items: center;
    /* aa只要三句话就可以实现不定宽高水平垂直居中。 */
}
.box>div{
    background: green;
    width: 200px;
    height: 200px;
}
```

px和em和rem的区别

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。

PX特点

1. IE无法调整那些使用px作为单位的字体大小；
2. 国外的大部分网站能够调整的原因在于其使用了em或rem作为字体单位；
3. Firefox能够调整px和em，rem，但是96%以上的中国网民使用IE浏览器(或内核)。

EM

em是相对长度单位。相对于当前对象内文本的字体尺寸。

EM特点

1. em的值并不是固定的；

2. em会继承父级元素的字体大小。

所以我们在写CSS的时候，需要注意两点：

1. body选择器中声明Font-size=62.5%；
2. 将你的原来的px数值除以10，然后换上em作为单位；
3. 重新计算那些被放大的字体的em数值。避免字体大小的重复声明。

REM

1、rem是相对大小，但相对的只是HTML根元素。

2、这个单位可谓集相对大小和绝对大小的优点于一身，通过它既可以做到只修改根元素就成比例地调整所有字体大小，又可以避免字体大小逐层复合的连锁反应。目前，除了IE8及更早版本外，所有浏览器均已支持rem。

3、对于不支持它的浏览器，应对方法也很简单，就是多写一个绝对单位的声明。

这些浏览器会忽略用rem设定的字体大小。下面就是一个例子：p {font-size: 14px; font-size:.875rem;}

px 与 rem 的选择？

对于只需要适配少部分手机设备，且分辨率对页面影响不大的，使用px即可。

对于需要适配各种移动设备，使用rem，例如只需要适配iPhone和iPad等分辨率差别比较挺大的设备。

rem如何处理不同手机屏幕的适配

<https://www.cnblogs.com/dannyxie/p/6640903.html>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  1  <meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale=1, minimum-scale=1">
  2  <script type="text/javascript">
    document.documentElement.style.fontSize = document.documentElement.clientWidth / 640 * 100 + 'px';
  3  </script>
</head>
<body style="margin: 0 ;padding: 0;font-size: 0.32rem">
  <div style="width: 3.2rem;height: 3.2rem ;background: red">
    <span>danny.xie</span>
  </div>
</body>
</html>
```

9、文本溢出样式

单行文本溢出

```
.box{
  overflow:hidden;
  text-overflow:ellipsis;
  white-space:nowrap;
}
```

多行文本溢出：

```
.box{
  display:-webkit-box;
  -webkit-box-orient:vertical;
  -webkit-line-clamp: 3;
  overflow:hidden;
}
```

CSS margin重叠问题

块元素在垂直方向上的margin是很奇怪的，会有重叠现象。

如果display都是block，有三种情况：

外间距均为正数，竖直方向上会选择最大的外边距作为间隔

一正一负，间距 = 正 - |负|

两个负，间距 = 0 - 绝对值最大的那个

设置display: inline-block的盒子不会有margin重叠，position: absolute的也不会出现。

10、如何解决移动端1px的问题，使用border-image的时候遇到圆角怎么办。

a、使用border-image实现

根据需求选择图片，然后根据css的border-image属性设置。代码如下

```
.border-image-1px {
  border-bottom: 1px solid #666;
}
@media only screen and (-webkit-min-device-pixel-ratio: 2) {
  .border-image-1px {
    border-bottom: none;
    border-width: 0 0 1px 0;
    -webkit-border-image: url(../img/linenew.png) 0 0 2 0 stretch;
    border-image: url(../img/linenew.png) 0 0 2 0 stretch;
  }
}
```

优点：可以设置单条、多条表框。缺点：更换颜色和样式麻烦，某些设备上会模糊。

使用border-image的时候遇到圆角怎么实现


```

.border{
  position: relative;
  border: 4px solid transparent;
  border-radius: 16px;
  background: linear-gradient(orange, violet);
  background-clip: padding-box;
  padding: 10px;
  /* just to show box-shadow still works fine */
  box-shadow: 0 3px 9px black, inset 0 0 9px white;
}
.border::after{
  position: absolute;
  top: -4px; bottom: -4px;
  left: -4px; right: -4px;
  background: linear-gradient(red, blue);
  content: '';
  z-index: -1;
  border-radius: 16px;
}

```

b、使用background-image实现

background-image 跟border-image的方法一样，你要先准备一张符合你要求的图片。优缺点与border-image一样。

```

.background-image-1px {
  background: url('../img/line.png') repeat-x left bottom;
  -webkit-background-size: 100% 1px; background-size: 100% 1px;
}

```

c、使用box-shadow模拟边框

代码如下

```

.box-shadow-1px {
  box-shadow: inset 0px -1px 1px -1px #c8c7cc;
}

```

优点：代码少，兼容性好。缺点：边框有阴影，颜色变浅。

d、伪类 + transform 实现

```
.scale-1px{
  position: relative;
  margin-bottom: 20px;
  border:none;
}
.scale-1px:after{
  content: '';
  position: absolute;
  top: 0;
  left: 0;
  border: 1px solid #000;
  -webkit-box-sizing: border-box;
  box-sizing: border-box;
  width: 200%;
  height: 200%;
  -webkit-transform: scale(0.5);
  transform: scale(0.5);
  -webkit-transform-origin: left top;
  transform-origin: left top;
}
```

优点：可以满足所有场景，且修改灵活。缺点：对于已使用伪类的元素要多层嵌套。

line-height的取值为整数和百分比的区别

line-height属性的细节

与大多数CSS属性不同，line-height支持属性值设置为无单位的数字。有无单位在子元素继承属性时有微妙的不同。

语法

line-height: normal | <number> | <length> | <percentage>

normal 根据浏览器决定，一般为1.2。

number 仅指定数字时（无单位），实际行距为字号乘以该数字得出的结果。可以理解为一个系数，子元素仅继承该系数，子元素的真正行距是分别与自身元素字号相乘的计算结果。大多数情况下推荐使用，可以避免一些意外的继承问题。

length 具体的长度，如px/em等。

percentage 百分比，100%与1em相同。

有单位（包括百分比）与无单位之间的区别

有单位时，子元素继承了父元素计算得出的行距；无单位时继承了系数，子元素会分别计算各自行距（推荐使用）。

Css3的动画和js的动画有什么区别

JS动画

缺点：

(1)JavaScript在浏览器的主线程中运行，而主线程中还有其它需要运行的JavaScript脚本、样式计算、布局、绘制任务等,对其干扰导致线程可能出现阻塞，从而造成丢帧的情况。

(2)代码的复杂度高于CSS动画

优点：

(1)JavaScript动画控制能力很强,可以在动画播放过程中对动画进行控制：开始、暂停、回放、终止、取消都是可以做到的。

(2)动画效果比css3动画丰富,有些动画效果，比如曲线运动,冲击闪烁,视差滚动效果，只有JavaScript动画才能完成

(3)CSS3有兼容性问题，而JS大多时候没有兼容性问题

CSS动画

缺点：

(1)运行过程控制较弱,无法附加事件绑定回调函数。CSS动画只能暂停,不能在动画中寻找一个特定的时间点，不能在半路反转动画，不能变换时间尺度，不能在特定的位置添加回调函数或是绑定回放事件,无进度报告


(2)代码冗长。想用 CSS 实现稍微复杂一点动画,最后CSS代码都会变得非常笨重。


优点：(1)浏览器可以对动画进行优化。


实现一个div,左边固定div宽度200px,右边div自适应




```
<div class="container">
  <div class="left">左边固定宽度</div>
  <div class="right">右边自适应剩余宽度</div>
</div>
```




/*方法一： BFC(块级格式化上下文)*/

```
.container {
  width: 1000px;
  height: 400px;
  border: 1px solid  red;
}



.left {
  width: 200px;
  height: 100%;
  background:  gray;
  float: left;
}




.rigth {
  overflow: hidden;
  /* 触发bfc */
  background:  green;
}
```

```
/*方法二： flex布局 */
.container {
    width: 1000px;
    height: 400px;
    border: 1px solid  red;
    display: flex;
    /*flex布局*/
}
.left {
    width: 200px;
    height: 100%;
    background:  gray;
    flex: none;
}
.right {
    height: 100%;
    background:  green;
    flex: 1;
    /*flex布局*/
}
```

```
/* 方法三: table布局 */
.container {
    width: 1000px;
    height: 400px;
    border: 1px solid  red;
    display: table;
    /*table布局*/
}
.left {
    width: 200px;
    height: 100%;
    background:  gray;
    display: table-cell;
}
.right {
    height: 100%;
    background:  green;
    display: table-cell;
}
```



```
/*方法四*/
.container {
    width: 100%;
    height: auto;
    overflow: hidden;
}
.left {
    float: left;
    width: 200px;
    height: auto;
    min-height: 300px;
    background:  red;
}
.right {
    margin-left: 200px;
    width: 100%;
    background:  yellow;
}
```

```
/*方法五:  css计算宽度calc*/
.container {
    width: 1000px;
    height: 400px;
    border: 1px solid  red;
}
.left {
    width: 200px;
    height: 100%;
    background:  gray;
    float: left;
}
.right {
    height: 100%;
    background:  green;
    float: right;
    width: calc(100% - 200px);
}
```