### 盒子模型

.box1{

使用width来设置盒子内容区的宽度

使用height来设置盒子内容区的高度

width和height只是设置的盒子内容区的大小，而不是盒子的整个大小，

盒子可见框的大小由内容区，内边距和边框共同决定

width: 300px;

height: 300px;

设置背景颜色

background-color: #bfa;

为元素设置边框

要为一个元素设置边框必须指定三个样式

border-width:边框的宽度

border-color:边框颜色

border-style:边框的样式

设置边框的宽度

border-width:10px ;

使用border-width可以分别指定四个边框的宽度

如果在border-width指定了四个值，

则四个值会分别设置给 上 右 下 左，按照顺时针的方向设置的

如果指定三个值，

则三个值会分别设置给 上 左右 下

如果指定两个值，

则两个值会分别设置给 上下 左右

如果指定一个值，则四边全都是该值

除了border-width，CSS中还提供了四个border-xxx-width

xxx的值可能是top right bottom left

专门用来设置指定边的宽度

border-width:10px 20px 30px 40px ;

border-width:10px 20px 30px ;

border-width: 10px 20px ;

border-width: 10px;

border-left-width:100px ;

设置边框的颜色

和宽度一样，color也提供四个方向的样式，可以分别指定颜色

border-xxx-color

border-color: red;

border-color: red yellow orange blue;

border-color: red yellow orange;

border-color: red yellow;

设置边框的样式

可选值：

none，默认值，没有边框

solid 实线

dotted 点状边框

dashed 虚线

double 双线

style也可以分别指定四个边的边框样式，规则和width一致，

同时它也提供border-xxx-style四个样式，来分别设置四个边

border-style: double;

border-style: solid dotted dashed double;

}

### 01.边框

.box{

width: 200px;

height: 200px;

background-color: #bfa;

设置边框

大部分的浏览器中，边框的宽度和颜色都是有默认值，而边框的样式默认值都是none

border-width:10px ;

border-color: red;

border-style: solid;

border

- 边框的简写样式，通过它可以同时设置四个边框的样式，宽度，颜色

- 而且没有任何的顺序要求

- border一指定就是同时指定四个边不能分别指定

border-top border-right border-bottom border-left

可以单独设置四个边的样式，规则和border一样，只不过它只对一个边生效

border: red solid 10px ;

border-left: red solid 10px ;

border-top: red solid 10px;

border-bottom: red solid 10px;

border-left: red solid 10px;

border: red solid 10px;

border-right: none;

}

### 02.内边距

.box1{

width: 200px;

height: 200px;

background-color: #bfa;

设置边框

border: 10px red solid;

内边距（padding），指的是盒子的内容区与盒子边框之间的距离

一共有四个方向的内边距，可以通过：

padding-top

padding-right

padding-bottom

padding-left

来设置四个方向的内边距

内边距会影响盒子的可见框的大小，元素的背景会延伸到内边距,

盒子的大小由内容区、内边距和边框共同决定

盒子可见框的宽度 = border-left-width + padding-left + width + padding-right + border-right-width

可见宽的高度 = border-top-width + padding-top + height + padding-bottom + border-bottom-width

设置上内边距

padding-top: 100px;

设置右内边距

padding-right: 100px;

padding-bottom: 100px;

padding-left: 100px;

使用padding可以同时设置四个边框的样式，规则和border-width一致

padding: 100px;

padding: 100px 200px;

padding: 100px 200px 300px;

padding: 100px 200px 300px 400px;

}

创建一个子元素box1占满box2

.box2{

width: 100%;

height: 100%;

background-color: yellow;

}

### 03.外边距

.box1{

width: 200px;

height: 200px;

background-color: #bfa;

border: 10px solid red;

外边距指的是当前盒子与其他盒子之间的距离，

他不会影响可见框的大小，而是会影响到盒子的位置。

盒子有四个方向的外边距：

margin-top

margin-right

margin-bottom

margin-left

由于页面中的元素都是靠左靠上摆放的，

所以注意当我们设置上和左外边距时，会导致盒子自身的位置发生改变，

而如果是设置右和下外边距会改变其他盒子的位置

设置box1的上外边距，盒子上边框和其他的盒子的距离

margin-top: 100px;

左外边距

margin-left: 100px;

设置右和下外边距

margin-right: 100px;

margin-bottom: 100px;

外边距也可以指定为一个负值，

如果外边距设置的是负值，则元素会向反方向移动

margin-left: -150px;

margin-top: -100px;

margin-bottom: -100px;

margin-bottom: -100px;

margin还可以设置为auto，auto一般只设置给水平方向的margin

如果只指定，左外边距或右外边距的margin为auto则会将外边距设置为最大值

垂直方向外边距如果设置为auto，则外边距默认就是0

如果将left和right同时设置为auto，则会将两侧的外边距设置为相同的值，

就可以使元素自动在父元素中居中，所以我们经常将左右外边距设置为auto

以使子元素在父元素中水平居中

margin-left: auto;

margin-right: auto;

外边距同样可以使用简写属性 margin，可以同时设置四个方向的外边距,

规则和padding一样

margin: 0 auto;

}

.box2{

width: 200px;

height: 200px;

background-color: yellow;

}

### 04.外边距

.box1{

width: 100px;

height: 100px;

background-color: red;

为上边的元素设置一个下外边距

margin-bottom: 100px;

}

垂直外边距的重叠

- 在网页中相邻的垂直方向的外边距会发生外边距的重叠

所谓的外边距重叠指兄弟元素之间的相邻外边距会取最大值而不是取和

如果父子元素的垂直外边距相邻了，则子元素的外边距会设置给父元素

.box2{

width: 100px;

height: 100px;

background-color: green;

为下边的元素设置一个上外边距

margin-top: 100px;

}

.box3{

width: 200px;

height: 100px;

background-color: yellow;

为box3设置一个上边框

border-top: 1px red solid;

padding-top: 1px;

padding-top: 100px;

}

.box4{

width: 100px;

height: 100px;

background-color: yellowgreen;

为子元素设置一个上外边距，是子元素的位置下移

margin-top: 100px;

}

### 05.默认样式

浏览器为了在页面中没有样式时，也可以有一个比较好的显示效果，

所以为很多的元素都设置了一些默认的margin和padding，

而它的这些默认样式，正常情况下我们是不需要使用的。

所以我们往往在编写样式之前需要将浏览器中的默认的margin和padding统统的去掉

清除浏览器的默认样式

{

margin: 0;

padding: 0;

}

.box1{

width: 100px;

height: 100px;

background-color: #bfa;

}

p{

background-color: yellow;

}

### 06.内联元素的盒子

span{

background-color: #bfa;

}

.box1{

width: 100px;

height: 100px;

background-color: red;

}

.s1{

内容区、内边距 、边框 、外边距

内联元素不能设置width和height

width: 200px;

height: 200px;

设置水平内边距,内联元素可以设置水平方向的内边距

padding-left: 100px ;

padding-right: 100px ;

垂直方向内边距，内联元素可以设置垂直方向内边距，但是不会影响页面的布局

padding-top: 50px;

padding-bottom: 50px;

为元素设置边框,

内联元素可以设置边框，但是垂直的边框不会影响到页面的布局

border: 1px blue solid;

水平外边距

内联元素支持水平方向的外边距

margin-left:100px ;

margin-right: 100px;

内联元素不支持垂直外边距

margin-top: 200px;

margin-bottom: 200px;

}

.s2{

为右边的元素设置一个左外边距

水平方向的相邻外边距不会重叠，而是求和

margin-left: 100px;

}

### 07.display

a{

background-color: #bfa;

将一个内联元素变成块元素，

通过display样式可以修改元素的类型

可选值：

inline：可以将一个元素作为内联元素显示

block: 可以将一个元素设置块元素显示

inline-block：将一个元素转换为行内块元素

- 可以使一个元素既有行内元素的特点又有块元素的特点

既可以设置宽高，又不会独占一行

none: 不显示元素，并且元素不会在页面中继续占有位置

display: none;

为其设置一个宽和高

width: 500px;

height: 500px;

}

.box{

width: 100px;

height: 100px;

background-color: orange;

display: none;

使用该方式隐藏的元素，不会在页面中显示，并且不再占据页面的位置

display: none;

visibility

- 可以用来设置元素的隐藏和显示的状态

- 可选值：

visible 默认值，元素默认会在页面显示

hidden 元素会隐藏不显示

使用 visibility:hidden;隐藏的元素虽然不会在页面中显示，

但是它的位置会依然保持

visibility:hidden ;

}

### 08.overflow

.box1{

width: 200px;

height: 200px;

background-color: #bfa;

子元素默认是存在于父元素的内容区中，

理论上讲子元素的最大可以等于父元素内容区大小

如果子元素的大小超过了父元素的内容区，则超过的大小会在父元素以外的位置显示，

超出父元素的内容，我们称为溢出的内容

父元素默认是将溢出内容，在父元素外边显示，

通过overflow可以设置父元素如何处理溢出内容：

可选值：

- visible，默认值，不会对溢出内容做处理，元素会在父元素以外的位置显示

- hidden, 溢出的内容，会被修剪，不会显示

- scroll, 会为父元素添加滚动条，通过拖动滚动条来查看完整内容

- 该属性不论内容是否溢出，都会添加水平和垂直双方向的滚动条

- auto，会根据需求自动添加滚动条，

需要水平就添加水平

需要垂直就添加垂直

都不需要就都不加

overflow: auto;

}

.box2{

width: 100px;

height: 500px;

background-color: red;

}

### 09.文档流

<!--

文档流

文档流处在网页的最底层，它表示的是一个页面中的位置，

我们所创建的元素默认都处在文档流中

元素在文档流中的特点

块元素

1.块元素在文档流中会独占一行，块元素会自上向下排列。

2.块元素在文档流中默认宽度是父元素的100%

3.块元素在文档流中的高度默认被内容撑开

内联元素

1.内联元素在文档流中只占自身的大小，会默认从左向右排列，

如果一行中不足以容纳所有的内联元素，则换到下一行，

继续自左向右。

2.在文档流中，内联元素的宽度和高度默认都被内容撑开

-->

<!--

当元素的宽度的值为auto时，此时指定内边距不会影响可见框的大小，

而是会自动修改宽度，以适应内边距

-->

### 10.浮动

.box1{

width: 600px;

height: 200px;

background-color: red;

块元素在文档流中默认垂直排列，所以这个三个div自上至下依次排开，

如果希望块元素在页面中水平排列，可以使块元素脱离文档流

使用float来使元素浮动，从而脱离文档流

可选值：

none，默认值，元素默认在文档流中排列

left，元素会立即脱离文档流，向页面的左侧浮动

right，元素会立即脱离文档流，向页面的右侧浮动

当为一个元素设置浮动以后（float属性是一个非none的值），

元素会立即脱离文档流，元素脱离文档流以后，它下边的元素会立即向上移动

元素浮动以后，会尽量向页面的左上或这是右上漂浮，

直到遇到父元素的边框或者其他的浮动元素

如果浮动元素上边是一个没有浮动的块元素，则浮动元素不会超过块元素

浮动的元素不会超过他上边的兄弟元素，最多最多一边齐

float: left;

}

.box2{

width: 600px;

height: 200px;

background-color: yellow;

float: left;

}

.box3{

width: 200px;

height: 200px;

background-color: green;

float: right;

}

{

margin: 0;

padding: 0;

}

.box1{

width: 100px;

height: 100px;

background-color: #bfa;

浮动的元素不会盖住文字，文字会自动环绕在浮动元素的周围，

所以我们可以通过浮动来设置文字环绕图片的效果

box1向左浮动

float: left;

}

.p1{

background-color: yellow;

}

.box1{

在文档流中，子元素的宽度默认占父元素的全部

background-color: #bfa;

当元素设置浮动以后，会完全脱离文档流.

块元素脱离文档流以后，高度和宽度都被内容撑开

float: left;

}

.s1{

开启span的浮动

内联元素脱离文档流以后会变成块元素

float: left;

width: 100px;

height: 100px;

background-color: yellow;

}

### 13.简单布局

清除默认样式

{

margin: 0;

padding: 0;

}

设置头部div

.header{

设置一个宽度

width: 1000px;

设置一个高度

height: 120px;

设置一个背景颜色

background-color: yellowgreen;

设置居中

margin: 0 auto;

}

设置一个content

.content{

设置一个宽度

width: 1000px;

设置一个高度

height: 400px;

设置一个背景颜色

background-color: orange;

居中

margin: 10px auto;

}

设置content中小div的样式

.left{

width: 200px;

height: 100%;

background-color: skyblue;

向左浮动

float: left;

}

.center{

width: 580px;

height: 100%;

background-color: yellow;

向左浮动

float: left;

设置水平外边距

margin: 0 10px;

}

.right{

width: 200px;

height: 100%;

background-color: pink;

向左浮动

float: left;

}

设置一个footer

.footer{

设置一个宽度

width: 1000px;

设置一个高度

height: 120px;

设置一个背景颜色

background-color: silver;

居中

margin: 0 auto;

}

### 01.高度塌陷

.box1{

为box1设置一个边框

border: 10px red solid;

}

.box2{

width: 100px;

height: 100px;

background-color: blue;

在文档流中，父元素的高度默认是被子元素撑开的，

也就是子元素多高，父元素就多高。

但是当为子元素设置浮动以后，子元素会完全脱离文档流，

此时将会导致子元素无法撑起父元素的高度，导致父元素的高度塌陷。

由于父元素的高度塌陷了，则父元素下的所有元素都会向上移动，这样将会导致页面布局混乱。

所以在开发中一定要避免出现高度塌陷的问题,

我们可以将父元素的高度写死，以避免塌陷的问题出现，

但是一旦高度写死，父元素的高度将不能自动适应子元素的高度，所以这种方案是不推荐使用的。

为子元素设置向左浮动

float: left;

}

.box3{

height: 100px;

background-color: yellow;

}

### 02.解决高度塌陷

.box1{

border: 10px red solid;

根据W3C的标准，在页面中元素都一个隐含的属性叫做Block Formatting Context

简称BFC，该属性可以设置打开或者关闭，默认是关闭的。

当开启元素的BFC以后，元素将会具有如下的特性：

1.父元素的垂直外边距不会和子元素重叠

2.开启BFC的元素不会被浮动元素所覆盖

3.开启BFC的元素可以包含浮动的子元素

如何开启元素的BFC

1.设置元素浮动

- 使用这种方式开启，虽然可以撑开父元素，但是会导致父元素的宽度丢失

而且使用这种方式也会导致下边的元素上移，不能解决问题

2.设置元素绝对定位

3.设置元素为inline-block

- 可以解决问题，但是会导致宽度丢失，不推荐使用这种方式

4.将元素的overflow设置为一个非visible的值

推荐方式：将overflow设置为hidden是副作用最小的开启BFC的方式。

overflow: hidden;

但是在IE6及以下的浏览器中并不支持BFC，所以使用这种方式不能兼容IE6。

在IE6中虽然没有BFC，但是具有另一个隐含的属性叫做hasLayout，

该属性的作用和BFC类似，所在IE6浏览器可以通过开hasLayout来解决该问题

开启方式很多，我们直接使用一种副作用最小的：

直接将元素的zoom设置为1即可

zoom表示放大的意思，后边跟着一个数值，写几就将元素放大几倍

zoom:1表示不放大元素，但是通过该样式可以开启hasLayout

zoom这个样式，只在IE中支持，其他浏览器都不支持

zoom:1;

overflow: hidden;

}

.box2{

width: 100px;

height: 100px;

background-color: blue;

float: left;

}

.box3{

height: 100px;

background-color: yellow;

}

### 03.导航条

清除默认样式

{

margin:0;

padding:0;

}

设置ul

.nav{

去除项目符号

list-style: none;

为ul设置一个背景颜色

background-color: #6495ED;

设置一个宽度

在IE6中，如果为元素指定了一个宽度，则会默认开启hasLayout

width: 1000px;

设置元素居中

margin: 50px auto;

解决高度塌陷

overflow: hidden;

}

设置li

.nav li{

设置li向左浮动

float: left;

width: 12.5%;

}

.nav a{

将a转换为块元素

display: block;

为a指定一个宽度

width: 100%;

设置文字居中

text-align: center;

设置一个上下内边距

padding: 5px 0;

去除下划线

text-decoration: none;

设置字体颜色

color: white;

设置加粗

font-weight: bold;

}

设置a的鼠标移入的效果

.nav a:hover{

background-color: #c00;

}

### 04.清除浮动

.box1{

width: 100px;

height: 100px;

background-color: yellow;

设置box1向左浮动

float: left;

}

.box2{

width: 200px;

height: 200px;

background-color: yellowgreen;

由于受到box1浮动的影响，box2整体向上移动了100px

我们有时希望清除掉其他元素浮动对当前元素产生的影响，这时可以使用clear来完成功能

clear可以用来清除其他浮动元素对当前元素的影响

可选值：

none，默认值，不清除浮动

left，清除左侧浮动元素对当前元素的影响

right，清除右侧浮动元素对当前元素的影响

both，清除两侧浮动元素对当前元素的影响

清除对他影响最大的那个元素的浮动

清除box1浮动对box2产生的影响

清除浮动以后，元素会回到其他元素浮动之前的位置

clear: left;

float: right;

}

.box3{

width: 300px;

height: 300px;

background-color: skyblue;

clear: both;

}

### 05.解决高度塌陷

.box1{

border: 1px solid red;

}

.box2{

width: 100px;

height: 100px;

background-color: blue;

float: left;

}

解决高度塌陷方案二：

可以直接在高度塌陷的父元素的最后，添加一个空白的div，

由于这个div并没有浮动，所以他是可以撑开父元素的高度的，

然后在对其进行清除浮动，这样可以通过这个空白的div来撑开父元素的高度，

基本没有副作用

使用这种方式虽然可以解决问题，但是会在页面中添加多余的结构。

.clear{

clear: both;

}

### 06.解决高度塌陷

.box1{

border: 1px solid red;

}

.box2{

width: 100px;

height: 100px;

background-color: blue;

float: left;

}

通过after伪类，选中box1的后边

可以通过after伪类向元素的最后添加一个空白的块元素，然后对其清除浮动，

这样做和添加一个div的原理一样，可以达到一个相同的效果，

而且不会在页面中添加多余的div，这是我们最推荐使用的方式，几乎没有副作用

.clearfix:after{

添加一个内容

content: "";

转换为一个块元素

display: block;

清除两侧的浮动

clear: both;

}

在IE6中不支持after伪类,

所以在IE6中还需要使用hasLayout来处理

.clearfix{

zoom:1;

}

### 07.开班信息

清除默认样式

{

margin: 0;

padding: 0;

}

统一页面中的字体

body{

font: 12px1 宋体;

}

设置outer的大小

.outer{

width: 300px;

height: 473px;

background-color: #bfa;

设置outer居中效果

margin: 50px auto;

}

设置title的边框

.title{

设置上边框

border-top:2px #019e8b solid ;

设置盒子的高度

height: 36px;

设置背景样式

background-color: #f5f5f5;

设置title的行高

line-height: 36px;

设置title的内边距

padding: 0px 22px 0px 16px;

}

设置title中的超链接

.title a{

float: right;

设置字体颜色

color: red;

}

设置h3

.title h3{

font: 16px36px "微软雅黑";

}

设置内容

.content{

border: 1px solid #deddd9;

设置内边距

padding: 0px 28px 0px 20px;

}

设置内容中的超链接

.content a{

color: black;

去除超链接的下划线

text-decoration: none;

设置字体大小

font-size: 12px;

}

为超链接添加一个hover伪类

.content a:hover{

color: red;

为超链接添加下划线

text-decoration: underline;

}

设置内容中的标题

.content h3{

margin-top: 14px;

margin-bottom: 16px;

}

设置右侧的a的样式

.content .right{

设置向右浮动

float: right;

}

设置ul的样式

.content ul{

去除项目符号

list-style: none;

为ul设置一个下边框

border-bottom: 1px dashed #deddd9;

}

取消最后一个ul的边框

.content .no-border{

border: none;

}

设置内容中的红色字体

.content .red{

color: red;

font-weight: bold;

}

设置内容中的li

.content li{

margin-bottom: 15px;

}

### 01.相对定位

.box1{

width: 200px;

height: 200px;

background-color: red;

}

.box2{

width: 200px;

height: 200px;

background-color: yellow;

定位：

- 定位指的就是将指定的元素摆放到页面的任意位置

通过定位可以任意的摆放元素

- 通过position属性来设置元素的定位

-可选值：

static：默认值，元素没有开启定位

relative：开启元素的相对定位

absolute：开启元素的绝对定位

fixed：开启元素的固定定位（也是绝对定位的一种）

当元素的position属性设置为relative时，则开启了元素的相对定位

1.当开启了元素的相对定位以后，而不设置偏移量时，元素不会发生任何变化

2.相对定位是相对于元素在文档流中原来的位置进行定位

3.相对定位的元素不会脱离文档流

4.相对定位会使元素提升一个层级

5.相对定位不会改变元素的性质，块还是块，内联还是内联

position: relative;

当开启了元素的定位（position属性值是一个非static的值）时，

可以通过left right top bottom四个属性来设置元素的偏移量

left：元素相对于其定位位置的左侧偏移量

right：元素相对于其定位位置的右侧偏移量

top：元素相对于其定位位置的上边的偏移量

bottom：元素相对于其定位位置下边的偏移量

通常偏移量只需要使用两个就可以对一个元素进行定位，

一般选择水平方向的一个偏移量和垂直方向的偏移量来为一个元素进行定位

left: 100px;

top: 200px;

left: 200px;

}

.box3{

width: 200px;

height: 200px;

background-color: yellowgreen;

}

.s1{

position: relative;

width: 200px;

height: 200px;

background-color: yellow;

}

### 02.绝对定位

.box1{

width: 200px;

height: 200px;

background-color: red;

}

.box2{

width: 200px;

height: 200px;

background-color: yellow;

当position属性值设置为absolute时，则开启了元素的绝对定位

绝对定位：

1.开启绝对定位，会使元素脱离文档流

2.开启绝对定位以后，如果不设置偏移量，则元素的位置不会发生变化

3.绝对定位是相对于离他最近的开启了定位的祖先元素进行定位的（一般情况，开启了子元素的绝对定位都会同时开启父元素的相对定位）

如果所有的祖先元素都没有开启定位，则会相对于浏览器窗口进行定位

4.绝对定位会使元素提升一个层级

5.绝对定位会改变元素的性质，

内联元素变成块元素，

块元素的宽度和高度默认都被内容撑开

position: absolute;

left: 100px;

top: 100px;

}

.box3{

width: 300px;

height: 300px;

background-color: yellowgreen;

}

.box4{

width: 300px;

height: 300px;

background-color: orange;

开启box4的相对定位

position: relative;

}

.s1{

width: 100px;

height: 100px;

background-color: yellow;

开启绝对定位

position: absolute;

}

### 03.固定定位

.box1{

width: 200px;

height: 200px;

background-color: red;

}

.box2{

width: 200px;

height: 200px;

background-color: yellow;

当元素的position属性设置fixed时，则开启了元素的固定定位

固定定位也是一种绝对定位，它的大部分特点都和绝对定位一样

不同的是：

固定定位永远都会相对于浏览器窗口进行定位

固定定位会固定在浏览器窗口某个位置，不会随滚动条滚动

IE6不支持固定定位

position: fixed;

left: 0px;

top: 0px;

}

.box3{

width: 200px;

height: 200px;

background-color: yellowgreen;

}

### 04.层级

.box1{

width: 200px;

height: 200px;

background-color: red;

position: relative;

z-index: 2;

opacity: 0.5;

filter: alpha(opacity=50);

}

.box2{

width: 200px;

height: 200px;

background-color: yellow;

开启绝对定位

position: absolute;

设置偏移量

top: 100px;

left: 100px;

如果定位元素的层级是一样，则下边的元素会盖住上边的

通过z-index属性可以用来设置元素的层级

可以为z-index指定一个正整数作为值，该值将会作为当前元素的层级

层级越高，越优先显示

对于没有开启定位的元素不能使用z-index

z-index: 25;

opacity: 0.5;

filter: alpha(opacity=50);

}

.box3{

width: 200px;

height: 200px;

background-color: yellowgreen;

position: relative;

z-index: 3;

position: absolute;

top: 200px;

left: 200px;

z-index: 30;

设置元素的透明背景

opacity可以用来设置元素背景的透明，

它需要一个0-1之间的值

0 表示完全透明

1 表示完全不透明

0.5 表示半透明

opacity: 0.5;

opacity属性在IE8及以下的浏览器中不支持

IE8及以下的浏览器需要使用如下属性代替

alpha(opacity=透明度)

透明度，需要一个0-100之间的值

0 表示完全透明

100 表示完全不透明

50 半透明

这种方式支持IE6，但是这种效果在IE Tester中无法测试

filter: alpha(opacity=50);

}

.box4{

width: 200px;

height: 200px;

background-color: orange;

开启相对定位

position: relative;

父元素的层级再高，也不会盖住子元素

z-index: 20;

top: 500px;

}

.box5{

width: 100px;

height: 100px;

background-color: skyblue;

开启绝对定位

position: absolute;

z-index: 10;

}

### 05.背景

.box1{

width: 1024px;

height: 724px;

margin: 0 auto;

设置背景样式

background-color: #bfa;

使用background-image来设置背景图片

- 语法：background-image:url(相对路径);

- 如果背景图片大于元素，默认会显示图片的左上角

- 如果背景图片和元素一样大，则会将背景图片全部显示

- 如果背景图片小于元素大小，则会默认将背景图片平铺以充满元素

可以同时为一个元素指定背景颜色和背景图片，

这样背景颜色将会作为背景图片的底色

一般情况下设置背景图片时都会同时指定一个背景颜色

background-image:url(img1.png);

background-repeat用于设置背景图片的重复方式

可选值：

repeat，默认值，背景图片会双方向重复（平铺）

no-repeat ，背景图片不会重复，有多大就显示多大

repeat-x， 背景图片沿水平方向重复

repeat-y，背景图片沿垂直方向重复

background-repeat: repeat-y;

}

### 06.导航条

.box1{

width: 990px;

height: 32px;

background-color: #bfa;

margin: 50px auto;

设置为背景图片

background-image: url(imgbg.gif);

设置水平方向重复

background-repeat: repeat-x;

}

### 07.背景

{

margin: 0;

padding: 0;

}

.box1{

height: 500px;

margin: 0 auto;

设置一个背景颜色

background-color: #bfa;

设置一个背景图片

background-image: url(img4.png);

设置一个图片不重复

background-repeat: no-repeat;

背景图片默认是贴着元素的左上角显示

通过background-position可以调整背景图片在元素中的位置

可选值：

该属性可以使用 top right left bottom center中的两个值

来指定一个背景图片的位置

top left 左上

bottom right 右下

如果只给出一个值，则第二个值默认是center

也可以直接指定两个偏移量，

第一个值是水平偏移量

- 如果指定的是一个正值，则图片会向右移动指定的像素

- 如果指定的是一个负值，则图片会向左移动指定的像素

第二个是垂直偏移量

- 如果指定的是一个正值，则图片会向下移动指定的像素

- 如果指定的是一个负值，则图片会向上移动指定的像素

background-position: -80px -40px;

background-attachment: fixed;

}

body{

background-image: url(img3.png);

background-repeat: no-repeat;

background-attachment用来设置背景图片是否随页面一起滚动

可选值：

scroll，默认值，背景图片随着窗口滚动

fixed，背景图片会固定在某一位置，不随页面滚动

不随窗口滚动的图片，我们一般都是设置给body，而不设置给其他元素

background-attachment:fixed ;

}

### 08.背景

body {

background-image: url(img2.jpg);

background-repeat: no-repeat;

background-position: center;

当背景图片的background-attachment设置为fixed时，

背景图片的定位永远相对于浏览器的窗口

background-attachment: fixed;

}

### 09.练习

做完功能以后，发现在第一次切换图片时，会发现图片有一个非常快的闪烁，

这个闪烁会造成一次不佳的用户体验。

产生问题的原因：

背景图片是以外部资源的形式加载进网页的，浏览器每加载一个外部资源就需要单独的发送一次请求，

但是我们外部资源并不是同时加载，浏览器会在资源被使用才去加载资源

我们这个练习，一上来浏览器只会加载link.png由于hover和active的状态没有马上触发，

所以hover.png和active.png并不是立即加载的

当hover被触发时，浏览器才去加载hover.png

当active被触发时，浏览器才去加载active.png

由于加载图片需要一定的时间，所以在加载和显示过程会有一段时间，背景图片无法显示，导致出现闪烁的情况

为了解决该问题，可以将三个图片整合为一张图片，这样可以同时将三张图片一起加载，就不会出现闪烁的问题了，

然后在通过background-position来切换要显示的图片的位置，这种技术叫做图片整合技术（CSS-Sprite）

优点：

1 将多个图片整合为一张图片里，浏览器只需要发送一次请求，可以同时加载多个图片，

提高访问效率，提高了用户体验。

2 将多个图片整合为一张图片，减小了图片的总大小，提高请求的速度，增加了用户体验

.btn:link{

将a转换为块元素

display: block;

设置宽高

width: 93px;

height: 29px;

设置背景图片

background-image: url(imgbtnbtn2.png);

设置背景图片不重复

background-repeat: no-repeat;

}

.btn:hover{

当是hover状态时，希望图片可以向左移动

background-position: -93px 0px;

}

.btn:active{

当是active状态时，希望图片再向左移动

background-position: -186px 0px;

}

### 10.雪碧图

.box1{

width: 129px;

height: 45px;

background-image:url(imgamazon-sprite\_.png) ;

}

.box2{

width: 42px;

height: 30px;

background-image:url(imgamazon-sprite\_.png) ;

设置偏移量

background-position: -58px -338px;

}

### 11.背景

body{

height: 5000px;

设置一个背景颜色

background-color: #bfa;

设置一个背景图片

background-image: url(img3.png);

设置背景不重复

background-repeat: no-repeat;

设置背景图片的位置

background-position: center center;

设置背景图片不随滚动条滚动

background-attachment: fixed;

background-color: #bfa;

background

- 通过该属性可以同时设置所有背景相关的样式

- 没有顺序的要求，谁在前睡在后都行

也没有数量的要求，不写的样式就使用默认值

background: #bfa url(img3.png) center center no-repeat fixed;

}

### 01.表格

<!--

表格在日常生活中使用的非常的多，比如excel就是专门用来创建表格的工具，

表格就是用来表示一些格式化的数据的，比如：课程表、银行对账单

在网页中也可以来创建出不同的表格。

-->

<!--

在HTML中，使用table标签来创建一个表格

-->

<table border="1" width="40%" align="center">

<!--

在table标签中使用tr来表示表格中的一行，有几行就有几个tr

-->

<tr>

<!-- 在tr中需要使用td来创建一个单元格，有几个单元格就有几个td -->

<td>A1<td>

<td>A2<td>

<td>A3<td>

<td>A4<td>

<tr>

<tr>

<td>B1<td>

<td>B2<td>

<td>B3<td>

<!--

rowspan用来设置纵向的合并单元格

-->

<td rowspan="2">B4<td>

<tr>

<tr>

<td>C1<td>

<td>C2<td>

<td>C3<td>

<tr>

<tr>

<td>D1<td>

<td>D2<td>

<!--

colspan横向的合并单元格

-->

<td colspan="2">D3<td>

<tr>

<table>

### 02.表格

设置表格的宽度

table{

width: 300px;

居中

margin: 0 auto;

边框

border:1px solid black;

table和td边框之间默认有一个距离

通过border-spacing属性可以设置这个距离

border-spacing:0px ;

border-collapse可以用来设置表格的边框合并

如果设置了边框合并，则border-spacing自动失效

border-collapse: collapse;

设置背景样式

background-color: #bfa;

}

设置边框

td , th{

border: 1px solid black;

}

设置隔行变色

tr:nth-child(even){

background-color: #bfa;

}

鼠标移入到tr以后，改变颜色

tr:hover{

background-color: #ff0;

}

### 03.长表格

<!--

有一些情况下表格是非常的长的，

这时就需要将表格分为三个部分，表头，表格的主体，表格底部

在HTML中为我们提供了三个标签：

thead 表头

tbody 表格主体

tfoot 表格底部

这三个标签的作用，就来区分表格的不同的部分，他们都是table的子标签，

都需要直接写到table中，tr需要写在这些标签当中

thead中的内容，永远会显示在表格的头部

tfoot中的内容，永远都会显示表格的底部

tbody中的内容，永远都会显示表格的中间

如果表格中没有写tbody，浏览器会自动在表格中添加tbody

并且将所有的tr都放到tbody中，所以注意tr并不是table的子元素，而是tbody的子元素

通过table > tr 无法选中行 需要通过tbody > tr

-->

### 04.表格的布局

<!--

以前表格更多的情况实际上是用来对页面进行布局的，但是这种方式早已被CSS所淘汰了

表格的列数由td最多的那行决定

表格是可以嵌套，可以在td中在放置一个表格

-->

### 05.完善clearfix

.box1{

width: 300px;

height: 300px;

background-color: #bfa;

}

.box2{

width: 200px;

height: 200px;

background-color: yellow;

子元素和父元素相邻的垂直外边距会发生重叠，子元素的外边距会传递给父元素

使用空的table标签可以隔离父子元素的外边距，阻止外边距的重叠

margin-top: 100px;

}

.box3{

border: 10px red solid;

}

.box4{

width: 100px;

height: 100px;

background-color: yellowgreen;

float: left;

}

解决父子元素的外边距重叠

.box1:before{

content: "";

display:table可以将一个元素设置为表格显示

display: table;

}

解决父元素高度塌陷

.clearfix:after{

content: "";

display: block;

clear: both;

}

经过修改后的clearfix是一个多功能的

既可以解决高度塌陷，又可以确保父元素和子元素的垂直外边距不会重叠

.clearfix:before,

.clearfix:after{

content: "";

display: table;

clear: both;

}

.clearfix{

zoom: 1;

}