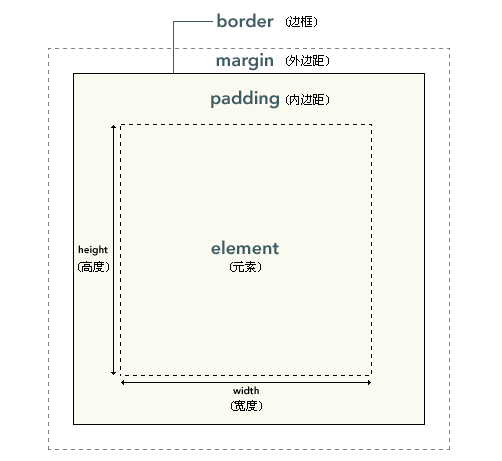
■框模型



元素框的最内部分是实际的内容，直接包围内容的是内边距。内边距呈现了元素的背景。内边距的边缘是边框。边框以外是外边距，外边距默认是透明的，因此不会遮挡其后的任何元素。

提示：背景应用于由内容和内边距、边框组成的区域。

■常用

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#dbox{

width: 100%;

height: 200px;

background-color: lawngreen;

}

#d1{

width: 100px;

height: 100px;

border: 1px solid black;

/\*设置左右剧中\*/

margin: 0 auto;

/\*鼠标滑过元素变小手\*/

cursor: pointer;

/\*设置圆角\*/

border-bottom-left-radius: 30px;

border-top-right-radius: 30px;

}

li{

/\*去除li前的圆点\*/

list-style: none;

}

a{

/\*去除a标签下划线\*/

text-decoration: none;

}

</style>

</head>

<body>

<div id="dbox">

<div id="d1"></div>

</div>

<ul>

<li>111</li>

<li>222</li>

<li>333</li>

</ul>

<a href="#">1212121212</a>

</body>

</html>

CSS3

█2D变换

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#d1{

width: 50px;

height: 50px;

background-color: lawngreen;

/\*设置过渡效果\*/

transition: width 2s, height 2s, transform 2s;

-moz-transition: width 2s, height 2s, transform 2s; /\* Firefox 4 \*/

-webkit-transition: width 2s, height 2s, transform 2s; /\* Safari 和 Chrome \*/

-o-transition: width 2s, height 2s, transform 2s; /\* Opera \*/

}

#d1:hover{

width: 100px;

height: 80px;

transform: rotate(30deg) translate(50px,100px);

-ms-transform: rotate(30deg) translate(50px,100px); /\* IE 9 \*/

-webkit-transform: rotate(30deg) translate(50px,100px); /\* Safari and Chrome \*/

-o-transform: rotate(30deg) translate(50px,100px); /\* Opera \*/

-moz-transform: rotate(30deg) translate(50px,100px); /\* Firefox \*/

/\*transform:scale(2,4)是宽增加2倍高增加4倍\*/

}

</style>

</head>

<body>

<div id="d1"></div>

</body>

</html>

█动画

先定义动画，设置一个动画名称。然后将该名称的动画运用到相应的元素中

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

/\*首先定义动画效果\*/

@keyframes myfirst

{

0% {background: red; left:0px; top:0px;}

25% {background: yellow; left:200px; top:0px;}

50% {background: blue; left:200px; top:200px;}

75% {background: green; left:0px; top:200px;}

100% {background: red; left:0px; top:0px;}

}

@-moz-keyframes myfirst /\* Firefox \*/

{

0% {background: red; left:0px; top:0px;}

25% {background: yellow; left:200px; top:0px;}

50% {background: blue; left:200px; top:200px;}

75% {background: green; left:0px; top:200px;}

100% {background: red; left:0px; top:0px;}

}

@-webkit-keyframes myfirst /\* Safari 和 Chrome \*/

{

0% {background: red; left:0px; top:0px;}

25% {background: yellow; left:200px; top:0px;}

50% {background: blue; left:200px; top:200px;}

75% {background: green; left:0px; top:200px;}

100% {background: red; left:0px; top:0px;}

}

@-o-keyframes myfirst /\* Opera \*/

{

0% {background: red; left:0px; top:0px;}

25% {background: yellow; left:200px; top:0px;}

50% {background: blue; left:200px; top:200px;}

75% {background: green; left:0px; top:200px;}

100% {background: red; left:0px; top:0px;}

}

/\*然后定义具体要动画的元素\*/

#d1{

width: 50px;

height: 50px;

/\*设置position才能使动画生效\*/

position: relative;

/\*设置动画\*/

animation: myfirst 5s;

-moz-animation: myfirst 5s; /\* Firefox \*/

-webkit-animation: myfirst 5s; /\* Safari 和 Chrome \*/

-o-animation: myfirst 5s; /\* Opera \*/

}

</style>

</head>

<body>

<div id="d1"></div>

</body>

</html>

如果不是如上面例子中详细设置动画各个事件点的动作，可以用下面的例子，只设定动画的开始和结束时间点

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

@keyframes myfirst1

{

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

@-moz-keyframes myfirst1 /\* Firefox \*/

{

from {

-moz-transform: rotate(0deg);

}

to {

-moz-transform: rotate(360deg);

}

}

@-webkit-keyframes myfirst1 /\* Safari 和 Chrome \*/

{

from {

-webkit-transform: rotate(0deg);

}

to {

-webkit-transform: rotate(360deg);

}

}

@-o-keyframes myfirst1 /\* Opera \*/

{

from {

-o-transform: rotate(0deg);

}

to {

-o-transform: rotate(360deg);

}

}

#d1{

width: 50px;

height: 50px;

background-color: lawngreen;

}

#d1:hover{

animation: myfirst1 1s;

-moz-animation: myfirst1 1s;

-webkit-animation: myfirst1 1s;

-o-animation: myfirst1 1s;

-webkit-animation-iteration-count: infinite;

-webkit-animation-timing-function: linear;

}

</style>

</head>

<body>

<div id="d1"></div>

</body>

</html>

█3D转换

原理同动画或过渡。

动画实现3D的例子：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

@keyframes myfirst1

{

from {

transform: rotateY(0deg);

}

to {

transform: rotateYG(180deg);

}

}

#d1:hover{

animation: myfirst1 1s;

-moz-animation: myfirst1 1s;

-webkit-animation: myfirst1 1s;

-o-animation: myfirst1 1s;

}

#d1{

width: 100px;

height: 50px;

background-color: lawngreen;

}

</style>

</head>

<body>

<div id="d1"></div>

</body>

</html>

过渡实现3D的例子

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#d1:hover{

transform: rotateY(180deg);

-ms-transform: rotateY(180deg); /\* IE 9 \*/

-webkit-transform: rotateY(180deg); /\* Safari and Chrome \*/

-o-transform: rotateY(180deg); /\* Opera \*/

-moz-transform: rotateY(180deg); /\* Firefox \*/

}

#d1{

width: 100px;

height: 50px;

background-color: lawngreen;

transition: transform 2s;

-moz-transition: transform 2s; /\* Firefox 4 \*/

-webkit-transition: transform 2s; /\* Safari 和 Chrome \*/

-o-transition: transform 2s; /\* Opera \*/

}

</style>

</head>

<body>

<div id="d1"></div>

</body>

</html>

●Transform的3D方法：

matrix3d(n,n,n,n,n,n,

n,n,n,n,n,n,n,n,n,n) 定义 3D 转换，使用 16 个值的 4x4 矩阵。

translate3d(x,y,z) 定义 3D 转化。

translateX(x) 定义 3D 转化，仅使用用于 X 轴的值。

translateY(y) 定义 3D 转化，仅使用用于 Y 轴的值。

translateZ(z) 定义 3D 转化，仅使用用于 Z 轴的值。

scale3d(x,y,z) 定义 3D 缩放转换。

scaleX(x) 定义 3D 缩放转换，通过给定一个 X 轴的值。

scaleY(y) 定义 3D 缩放转换，通过给定一个 Y 轴的值。

scaleZ(z) 定义 3D 缩放转换，通过给定一个 Z 轴的值。

rotate3d(x,y,z,angle) 定义 3D 旋转。

rotateX(angle) 定义沿 X 轴的 3D 旋转。

rotateY(angle) 定义沿 Y 轴的 3D 旋转。

rotateZ(angle) 定义沿 Z 轴的 3D 旋转。

perspective(n) 定义 3D 转换元素的透视视图。

█CSS3转换的原理总结

转换方式有过渡（transition）和动画（animation）

转换类型有2D和3D

Transform 2D:

函数 描述

matrix(n,n,n,n,n,n) 定义 2D 转换，使用六个值的矩阵。

translate(x,y) 定义 2D 转换，沿着 X 和 Y 轴移动元素。

translateX(n) 定义 2D 转换，沿着 X 轴移动元素。

translateY(n) 定义 2D 转换，沿着 Y 轴移动元素。

scale(x,y) 定义 2D 缩放转换，改变元素的宽度和高度。

scaleX(n) 定义 2D 缩放转换，改变元素的宽度。

scaleY(n) 定义 2D 缩放转换，改变元素的高度。

rotate(angle) 定义 2D 旋转，在参数中规定角度。

skew(x-angle,y-angle) 定义 2D 倾斜转换，沿着 X 和 Y 轴。

skewX(angle) 定义 2D 倾斜转换，沿着 X 轴。

skewY(angle) 定义 2D 倾斜转换，沿着 Y 轴。

Transform 3D:

函数 描述

matrix3d(n,n,n,n,n,n,

n,n,n,n,n,n,n,n,n,n) 定义 3D 转换，使用 16 个值的 4x4 矩阵。

translate3d(x,y,z) 定义 3D 转化。

translateX(x) 定义 3D 转化，仅使用用于 X 轴的值。

translateY(y) 定义 3D 转化，仅使用用于 Y 轴的值。

translateZ(z) 定义 3D 转化，仅使用用于 Z 轴的值。

scale3d(x,y,z) 定义 3D 缩放转换。

scaleX(x) 定义 3D 缩放转换，通过给定一个 X 轴的值。

scaleY(y) 定义 3D 缩放转换，通过给定一个 Y 轴的值。

scaleZ(z) 定义 3D 缩放转换，通过给定一个 Z 轴的值。

rotate3d(x,y,z,angle) 定义 3D 旋转。

rotateX(angle) 定义沿 X 轴的 3D 旋转。

rotateY(angle) 定义沿 Y 轴的 3D 旋转。

rotateZ(angle) 定义沿 Z 轴的 3D 旋转。

perspective(n) 定义 3D 转换元素的透视视图。

一般看具体情况，简单的就用过渡，复杂的就用动画。

█动画补充，设置不停重复播放动画

@keyframes myfirst2{

from {

             transform: rotate(0deg);

             }

             to {

             transform: rotate(360deg);

             }

}

#d3{

width:100px;

height: 100px;

background-color: lawngreen;

animation: myfirst2 1s;

/\* 以下是设置不停重复的播放动画效果 \*/

animation-iteration-count: infinite;

     animation-timing-function: linear;

}

█表格的几个常用设置

<style>

table{

border: 1px solid #e9e9e9;

/\* 使单元格之间没有间距 \*/

border-spacing: 0;

/\* 单元格之间边框重叠时只显示一条线 \*/

border-collapse: collapse;

}

tr, td, th{

border: 1px solid #e9e9e9;

/\* 设置内容不换行 \*/

white-space: nowrap;

/\* 空单元格的边框是否显示 \*/

empty-cells: show;

}

</style>