一：2D转换：

通过 CSS3 transform转换，我们能够对元素进行移动、缩放、转动、拉长或拉伸.

1. 2D移动：translate()。使用translate()函数，你可以把元素从原来的位置移动。移动参照元素左上角原点
   1. 语法：translate(tx) | translate(tx,ty)
   2. tx是一个代表X轴（横坐标）移动的向量长度，当其值为正值时，元素向X轴右方向移动，反之其值为负值时，元素向X轴左方向移动。
   3. ty是一个代表Y轴（纵向标）移动的向量长度，当其值为正值时，元素向Y轴下方向移动，反之其值为负值时，元素向Y轴上方向移动。如果ty没有显式设置时，相当于ty=0。
   4. 也可以使用translateX(tx) 或者 translateY(ty)
   5. 案例示例：

div:hover{  
 /\*设置两个值，第一个参数表示X方向 第二个参数表示Y方向\*/  
 /\*transform: translate(100px,100px);\*/  
 /\*也可以只传入一个参数，表示X方向\*/  
 /\*transform: translate(100px);\*/  
 /\*也可以指定具体的方向,如下代码，表示Y方向正值方向上移动100px\*/  
 transform:translateY(100px);  
}

1. 2D缩放：scale()：缩放scale()函数让元素根据中心原点对对象进行缩放。默认的值1。因此0.01到0.99之间的任何值，使一个元素缩小；而任何大于或等于1.01的值，让元素显得更大.缩放是参照元素中心点。
   1. 语法：scale(sx|ty) | scale(sx,sy)
   2. sx：用来指定横向坐标（X轴）方向的缩放向量，如果值为0.01~0.99之间，会让对象在X轴方向缩小，如果值大于或等于1.01，对象在Y轴方向放大。
   3. sy：用来指定纵向坐标（Y轴）方向的缩放量，如果值为0.01~0.99之间，会让对象在Y轴方向缩小，如果值大于或等于1.01，对象在Y轴方向放大
   4. 也可以使用scaleX(sx) 或者scaleY(sy)
   5. 案例示例：

div:hover{  
 /\*传入两个值，第一个参数表示X方向的缩放 第二个参数表示Y方向上的缩放\*/  
 /\*transform: scale(2,0.5);\*/  
 /\*也可以只传入一个值，表示X方向和Y方向上相同的缩放\*/  
 /\*transform: scale(2);\*/  
 /\*也可以指定具体方向上的缩放\*/  
 transform: scaleX(2);  
}

1. 2D旋转：rotate()：旋转rotate()函数通过指定的角度参数对元素根据对象原点指定一个2D旋转。它主要在二维空间内进行操作，接受一个角度值，用来指定旋转的幅度。如果这个值为正值，元素相对原点中心顺时针旋转；如果这个值为负值，元素相对原点中心逆时针旋转
   1. 语法：rotate(a);
   2. a: 代表的是一个旋转的角度值。其取值可以是正的，也可以是负的。如果取值为正值时，元素默认之下相对元素中心点顺时针旋转；如果取值为负值时，元素默认之下相对元素中心点逆时针旋转
   3. 案例示例：

div:hover{  
 /\*传入旋转的角度，如果正值，则进行顺时针旋转\*/  
 /\*transform: rotate(90deg);\*/  
 /\*如果传入负值，则逆时针旋转\*/  
 transform: rotate(-270deg);  
}

1. 2D翻转：skew()：能够让元素倾斜显示。它可以将一个对象以其中心位置围绕着X轴和Y轴按照一定的角度倾斜。这与rotate()函数的旋转不同，rotate()函数只是旋转，而不会改变元素的形状。skew()函数不会旋转，而只会改变元素的形状
   1. 语法：skew(ax) | skew(ax,ay)
   2. ax：用来指定元素水平方向（X轴方向）倾斜的角度。
   3. ay：用来指定元素垂直方向（Y轴方向）倾斜的角度。如果未显式的设置这个值，其默认为０。
   4. 也可以使用skewX(sx) 或者 skewY(sy)
   5. 案例示例：

div:hover{  
 /\*在X方向上倾斜30度\*/  
 transform: skewX(30deg);  
}

1. transform-origin: 允许你改变被转换元素的位置。
   1. 示例：

div{  
 width: 100px;  
 height: 100px;  
 margin: 100px auto;  
 background-color: red;  
 /\*添加过渡\*/  
 transition:all .5s;  
 /\*设置缩放的中心，默认是元素中心位置,现修改为元素左上角\*/  
 transform-origin: 0px 0px;  
}  
div:hover{  
 transform: scale(2);  
}

二：3D转换：

三维变换使用基于二维变换的相同属性，可以让我们基于三个坐标方向对元素进行变换。和二维变形一样，三维变形可以使用transform属性来设置

1. 3D移动
   1. 方法：translate3d(x,y,z) 使元素在这三个纬度中移动,也可以分开写，如：translateX(length),translateY(length), translateZ(length)
   2. 示例：

div:hover{  
 /\*Y轴移动+100px\*/  
 /\*transform:translateY(100px);\*/  
 /\*X轴移动100px\*/  
 /\*transform:translateX(100px);\*/  
 /\*x轴和Y轴方向同时移动\*/  
 transform:translate3d(100px,100px,0px);  
}

1. 3D缩放
   1. scale3d(number,number,number) 使元素在这三个纬度中缩放，也可分开写，如：scaleX(),scaleY(),scaleZ()
   2. 示例

div:hover{  
 /\*Y轴方向放大1倍\*/  
 /\*transform: scaleX(2);\*/  
 /\*X轴方向缩小0.5\*/  
 /\*transform: scaleX(0.5);\*/  
 /\*x轴和Y轴方向同时进行缩放\*/  
 transform: scale3d(2,0.5,1);  
}

1. 3D旋转
2. rotate3d(x,y,z,angle)：指定需要进行旋转的坐标轴
3. rotateX(angle) 是元素依照x轴旋转；
4. rotateY(angle) 是元素依照y轴旋转；
5. rotateZ(angle) 是元素依照z轴旋转
6. 示例：

div:hover{  
 /\*Y轴方向旋转45度\*/  
 /\*transform: rotateY(45deg);\*/  
 /\*X轴方向旋转90度\*/  
 /\*transform: rotateX(90deg);\*/  
 /\*x轴和Y轴方向同时进行旋转放\*/  
 transform: rotate3d(1,1,0,45deg);  
}

1. 透视/景深效果

左手法则：大拇指指向当前坐标轴的下方向，手指环绕的方向就是正方向

1. perspective(length) 为一个元素设置三维透视的距离。仅作用于元素的后代，而不是其元素本身。当perspective:none/0;时，相当于没有设perspective(length)。比如你要建立一个小立方体，长宽高都是200px。如果你的perspective < 200px ，那就相当于站在盒子里面看的结果，如果perspective 非常大那就是站在非常远的地方看（立方体已经成了小正方形了），意味着perspective 属性指定了观察者与z=0平面的距离，使具有三维位置变换的元素产生透视效果
2. perspective-origin属性规定了镜头在平面上的位置。默认是放在元素的中心
3. transform-style：使被转换的子元素保留其 3D 转换(需要设置在父元素中)：

|  |  |
| --- | --- |
| **值** | **描述** |
| flat | 子元素将不保留其 3D 位置-平面方式。 |
| preserve-3d | 子元素将保留其 3D 位置—立体方式。 |

1. 案例：立方体

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <style>  
 \*{  
 padding: 0;  
 margin: 0;  
 }  
 .box{  
 width: 200px;  
 height: 200px;  
 margin:100px auto;  
 position: relative;  
 opacity: 0.5;  
 /\*设置一下默认的变换，否则由于观察角度在正面，无法看到立方体\*/  
 transform: rotateX(-30deg) rotateY(30deg);  
 /\*保持3D效果\*/  
 transform-style: preserve-3d;  
 }  
 /\*公共样式\*/  
 .front,.back,.left,.right,.top,.bottom{  
 width: 200px;  
 height: 200px;  
 position: absolute;  
 left: 0;  
 top: 0;  
 }  
 .left{  
 background: red;  
 transform: translateX(-100px) rotateY(-90deg);  
 }  
 .right{  
 background: green;  
 transform: translateX(100px) rotateY(90deg);  
 }  
 .front{  
 background: blue;  
 transform: translateZ(100px);  
 }  
 .back{  
 background: pink;  
 transform: translateZ(-100px);  
 }  
 .top{  
 background: orange;  
 transform: translateY(-100px) rotateX(90deg);  
 }  
 .bottom{  
 background: purple;  
 transform: translateY(100px) rotateX(-90deg);  
 }  
 </style>  
</head>  
<body>  
<div class="box">  
 <div class="front">front</div>  
 <div class="back">back</div>  
 <div class="left">left</div>  
 <div class="right">right</div>  
 <div class="top">top</div>  
 <div class="bottom">bottom</div>  
</div>  
</body>  
</html>

三：动画：

动画是CSS3中具有颠覆性的特征之一，可通过设置多个节点来精确控制一个或一组动画，常用来实现复杂的动画效果.

1. 必要元素：

a、通过@keyframes指定动画序列；自动补间动画，确定两个点，系统会自动计算中间过程。这两个点就称为关键帧。我们可以设置多个关键帧

b、通过百分比将动画序列分割成多个节点；

c、在各节点中分别定义各属性

d、通过animation将动画应用于相应元素；

1. animation样式常用属性：
2. 动画序列的名称:animation-name: move;
3. 动画的持续时间:animation-duration: 1s;
4. 动画的延时:animation-delay: 1s;
5. **播放状态:animation-play-state: paused|running;**
6. 播放速度:animation-timing-function: linear;
7. 播放次数 反复:animation-iteration-count: 1;
8. 动画播放完结后的状态:animation-fill-mode: forwards;
9. 循环播放时，交叉动画:animation-direction: alternate;
10. 代码说明：

<style>  
 \*{  
 padding: 0;  
 margin: 0;  
 }  
 div{  
 width: 300px;  
 height: 300px;  
 margin:100px auto;  
 }  
 div > img{  
 width:100%;  
 }  
 /\*添加动画\*/  
 @keyframes rotateAni {  
 0%{  
 /\*可以同时对多个属性添加动画效果\*/  
 transform: rotate(0deg) scale(1);  
 }  
 50%{  
 transform: rotate(180deg) scale(2);  
 }  
 100%{  
 transform: rotate(360deg) scale(1);  
 }  
 }  
 div:hover > img{  
 /\*动画名称-自定义\*/  
 animation-name: rotateAni;  
 /\*动画时间\*/  
 animation-duration: 1s;  
 /\*动画速率曲线： linear：匀速 ease：动画以低速开始，然后加快，在结束前变慢 ease-in：动画以低速开始 ease-out：动画以低速结束 ease-in-out：动画以低速开始和结束\*/  
 animation-timing-function: linear;  
 /\*动画播放次数\*/  
 animation-iteration-count: 4;  
 /\*动画时间延迟\*/  
 animation-delay: 0s;  
 /\*动画播放完的状态： forwards:保持动画播放完毕后的状态 backwards:退回到原始状态(默认值)\*/  
 animation-fill-mode: forwards;  
 /\*动画是否轮流反射播放： alternate:在规定的次数内轮流反射播放 normal:正常播放\*/  
 /\*animation-direction: alternate;\*/  
 }  
 div:active >img{  
 /\*动画的当前播放状态： paused：暂停 running:运行\*/  
 animation-play-state: paused;  
 }  
</style>

1. 动画案例：汽车/钟表/宇宙

四：web字体和字体图标：

开发人员可以为自已的网页指定特殊的字体，无需考虑用户电脑上是否安装了此特殊字体，从此把特殊字体处理成图片的时代便成为了过去。它的支持程度比较好，甚至IE低版本浏览器也能支持。

1. 字体格式

不同浏览器所支持的字体格式是不一样的，我们有必要了解一下有关字体格式的知识。

1、TureTpe(.ttf)格式

.ttf字体是Windows和Mac的最常见的字体，是一种RAW格式，支持这种字体的浏览器有IE9+、Firefox3.5+、Chrome4+、Safari3+、Opera10+、iOS Mobile、Safari4.2+；

2、OpenType(.otf)格式

.otf字体被认为是一种原始的字体格式，其内置在TureType的基础上，支持这种字体的浏览器有Firefox3.5+、Chrome4.0+、Safari3.1+、Opera10.0+、iOS Mobile、Safari4.2+；

3、Web Open Font Format(.woff)格式

woff字体是Web字体中最佳格式，他是一个开放的TrueType/OpenType的压缩版本，同时也支持元数据包的分离，支持这种字体的浏览器有IE9+、Firefox3.5+、Chrome6+、Safari3.6+、Opera11.1+；

4、Embedded Open Type(.eot)格式

.eot字体是IE专用字体，可以从TrueType创建此格式字体，支持这种字体的浏览器有IE4+；

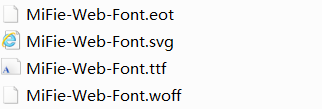
5、SVG(.svg)格式

.svg字体是基于SVG字体渲染的一种格式，支持这种字体的浏览器有Chrome4+、Safari3.1+、Opera10.0+、iOS Mobile Safari3.2+

1. 字体图标

常见的是把网页常用的一些小的图标，借助工具帮我们生成一个字体包，然后就可以像使用文字一样使用图标了

* + - 1. 优点
         * 将所有图标打包成字体库，减少请求；
         * 具有矢量性，可保证清晰度；
         * 使用灵活，便于维护
      2. 生成字体图标文件
         1. 下载字体图标文件：如https://www.awesomes.cn/的网站的介绍和使用



* + - * 1. 自定义字体

/\*定义字体图标\*/  
@font-face {  
 font-family: 'wjs'; //自定义的字体名称  
 src: url('../fonts/MiFie-Web-Font.eot'); /\* IE9\*/  
 src: url('../fonts/MiFie-Web-Font.eot') format('embedded-opentype'), /\* IE6-IE8 \*/  
 url('../fonts/MiFie-Web-Font.woff') format('woff'), /\* chrome、firefox \*/  
 url('../fonts/MiFie-Web-Font.ttf') format('truetype'), /\* chrome、firefox、opera、Safari, Android, iOS 4.2+\*/  
 url('../fonts/MiFie-Web-Font.svg') format('svg'); /\* iOS 4.1- \*/  
}

* + - * 1. 通过css样式使用字体

/\*自定义字体图标\*/  
.wjs\_font\_icon{  
 font-family: wjs; //这里对应着自定义的字体名称  
}  
/\*手机图标对应的编码\*/  
.wjs\_font\_icon\_phone::before{  
 content: "\e908"; //指定显示的内容  
}

五：多列布局

CSS3中新出现的多列布局(multi-column)是传统HTML网页中块状布局模式的有力扩充。这种新语法能够让WEB开发人员轻松的让文本呈现多列显示。我们知道，当一行文字太长时，读者读起来就比较费劲，有可能读错行或读串行；人们的视点从文本的一端移到另一端、然后换到下一行的行首，如果眼球移动浮动过大，他们的注意力就会减退，容易读不下去。所以，为了最大效率的使用大屏幕显示器，页面设计中需要限制文本的宽度，让文本按多列呈现，就像报纸上的新闻排版一样

1. 常用属性：
   1. column-count: 属性设置列的具体个数
   2. column-width: 属性控制列的宽度
   3. column-gap: 两列之间的缝隙间隔
   4. column-rule: 规定列之间的宽度、样式和颜色
   5. column-span: 规定元素应横跨多少列(n:指定跨n列 all:跨所有列)
2. 多列的用法：

<style>  
 \*{  
 padding: 0;  
 margin: 0;  
 }  
 .wrapper{  
 width:1054px;  
 padding:20px;  
 margin:0 auto;  
 font-family: "微软雅黑",Arial;  
 /\*设置以几列的方式显示\*/  
 column-count:2;  
 /\*指定列宽\*/  
 column-width:500px;  
 /\*指定列与列之间的间距\*/  
 column-gap: 50px;  
 /\*指定列与列之间间隙的样式\*/  
 /\*column-rule:2px dotted red\*/  
 /\*相对应下面的三个属性\*/  
 column-rule-color:red;  
 column-rule-style:dotted;  
 column-rule-width:2px;  
 }  
 .wrapper > h4{  
 column-span: all;  
 }  
</style>

1. 列高度的平衡：

如果设定列的最大高度，这个时候，文本内容会从第一列开始填充，然后第二列，第三列

max-height: 300px;

六：伸缩布局

布局的传统解决方案，基于[盒状模型](https://developer.mozilla.org/en-US/docs/Web/CSS/box_model)，依赖 [display](https://developer.mozilla.org/en-US/docs/Web/CSS/display)属性 + [position](https://developer.mozilla.org/en-US/docs/Web/CSS/position)属性 + [float](https://developer.mozilla.org/en-US/docs/Web/CSS/float)属性。它对于那些特殊布局非常不方便。CSS3在布局方面做了非常大的改进，使得我们对块级元素的布局排列变得十分灵活，适应性非常强，其强大的伸缩性，在响应式开中可以发挥极大的作用。

1. 重要属性
   1. display:flex:如果一个容器设置了这个属性，那么这个盒子里面的所有直接子元素都会自动的变成伸缩项(flex item)
   2. **j**ustify-content: 设置或检索弹性盒子元素在主轴（横轴）方向上的对齐方式,语法：justify-content：flex-start | flex-end | center | space-between | space-around

* flex-start：弹性盒子元素将向行起始位置对齐。该行的第一个子元素的主起始位置的边界将与该行的主起始位置的边界对齐，同时所有后续的伸缩盒项目与其前一个项目对齐。
* flex-end：弹性盒子元素将向行结束位置对齐。该行的第一个子元素的主结束位置的边界将与该行的主结束位置的边界对齐，同时所有后续的伸缩盒项目与其前一个项目对齐。
* center：弹性盒子元素将向行中间位置对齐。该行的子元素将相互对齐并在行中居中对齐，同时第一个元素与行的主起始位置的边距等同与最后一个元素与行的主结束位置的边距（如果剩余空间是负数，则保持两端相等长度的溢出）。
* space-between：弹性盒子元素会平均地分布在行里。如果最左边的剩余空间是负数，或该行只有一个子元素，则该值等效于'flex-start'。在其它情况下，第一个元素的边界与行的主起始位置的边界对齐，同时最后一个元素的边界与行的主结束位置的边距对齐，而剩余的伸缩盒项目则平均分布，并确保两两之间的空白空间相等。
* space-around：弹性盒子元素会平均地分布在行里，两端保留子元素与子元素之间间距大小的一半。如果最左边的剩余空间是负数，或该行只有一个伸缩盒项目，则该值等效于'center'。在其它情况下，伸缩盒项目则平均分布，并确保两两之间的空白空间相等，同时第一个元素前的空间以及最后一个元素后的空间为其他空白空间的一半。
  1. flex-flow: flex-flow属性是flex-direction属性和flex-wrap属性的简写形式，默认值为row nowrap。它用来设置或检索弹性盒模型对象的子元素排列方式,它的两种取值：flex-direction：定义弹性盒子元素的排列方向。flex-wrap：控制flex容器是单行或者多行。
     + flex-direction： **flex-direction**：row | row-reverse | column | column-reverse
* row：主轴与行内轴方向作为默认的书写模式。即横向从左到右排列（左对齐）。
* row-reverse：对齐方式与row相反。
* column：主轴与块轴方向作为默认的书写模式。即纵向从上往下排列（顶对齐）。
* column-reverse：对齐方式与column相反。
  + - flex-wrap： flex-wrap：nowrap | wrap | wrap-reverse
      * nowrap：flex容器为单行。该情况下flex子项可能会溢出容器
      * wrap：flex容器为多行。该情况下flex子项溢出的部分会被放置到新行，子项内部会发生断行
      * wrap-reverse：反转 wrap 排列。

1. flex属性：flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选
   * + - 语法：flex: [flex-grow] [flex-shrink] [flex-basis]：大多数情况下没必要使用这种语法，当使用flex缩写时，大多数情况下没必要使用这种语法
       - flex: [number]：这个语法指定了一个数字，代表了这个伸缩项目该占用的剩余空间比例
       - flex: auto：属性值被设为auto的伸缩项目，会根据主轴自动伸缩以占用所有剩余空间