接css3新特性

4.2. 2D转换

转换(transform)是css3中具有颠覆性的特性之一，可以实现元素的位移，旋转，缩放等效果

转换可以简单理解为变形

功能：

- 移动（transform)
- 旋转(rotate)
- 缩放(scale)

2D转换之移动(translate)

2D移动是2D转换里面的一种功能，可以改变元素在页面中的位置，类似定位

语法：

```
transform:translate(x,y);
```

或者分开写

```
transform:translateX(n);
trnasform:translateY(n);
```

重点：

- 定义2D转换中的移动，沿着X和Y轴移动元素
- translate最大的优点：不会影响到其他元素的位置
- translate中的百分比单位是相对于自身元素的translate:(50%,50%);
- 对行内标签没有效果

4.3. 2D转换之旋转rotate

2D旋转是让元素在2维平面内顺时针旋转或者逆时针旋转

语法：

```
transform:rotate(度数);
```

重点：

- rotate里面跟度数，单位是deg 比如rotate(45deg)
- 角度为正时，顺时针，负时，逆时针
- 默认旋转中心点是元素的中心点
- 案例：三角形

4.4. 2D转换中心点transform-origin

我们可以设置元素转换的中心点

语法：

```
transform-origin:x y;
```

重点：

- 注意后面的x和y用空格隔开
- x y默认转换的中心点是元素的中心点（50% 50%）
- 还可以给x y 设置像素或者方位名词（center top bottom right left）

4.5. 2D转换之缩放scale

语法：

```
transform:scale(x,y);
```

注意：

- 其中的x和y是用逗号分隔
- transform:scale(1,1)宽和高都放大一倍，相当于没有放大
- transform;scale(2,2)高和宽都放大了两倍
- transform:scale(2)只写了一个参数，第二个参数则和第一个参数一样，相当于scale(2,2)
- transform(0.5,0.5)缩小
- scale缩放的优势：可以设置转换中心点缩放，默认以中心点缩放的，而且不影响其他盒子

4.6. 2D转换的综合写法

注意：

- 如果同时使用多个转换，其格式为：transform:translate()rotate()scale()...等其顺序会影响转换的效果（先旋转会改变坐标轴方向）
- 当我们同时有位移和其他属性时，记得要将位移放到最前

5. 动画

动画(animation)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

相比较过渡，动画可以实现更多变化，更多控制，连续自动播放等效果

5.1. 动画的基本使用

制作动画分两步：

- 先定义动画
- 再使用（调用）动画

(1)先用keyframes定义动画（类似定义类选择器）

语法：

```
@keyframes 动画名称{
    0%{
        widht:100px;
    }
    100%{
        width:200px;
    }
}
```

动画序列：

- 0%是动画的开始，100%是动画的完成，这样的规则就是动画序列
- 在@keyframes中规定某项css样式，就能创建由当前样式逐渐改为新样式的动画效果
- 动画是使元素从一种样式逐渐转变为另一种样式的效果，可以改变任意多的样式任意多的次数
- 请用百分比来规定变化发生的时间，或用关键字“from”和“to”，等同于0%和100%

(2)元素使用动画

语法：

```
div{
    width:200px;
    height:200px;
    background-color:aqua;
    margin:100px auto;
    /*调用动画*/
    animation-name:动画名称;
    /*持续时间*/
    animation-duration:持续时间;
}
```

5.2. 动画常用属性

5.3. 动画简写属性

```
animation：动画名称 持续时间 运动曲线
何时开始 播放次数 是否反方向 动画起始或结束的状态
```

- 持续时间和何时开始都是一个时间，当两个属性都有时，持续时间要在前面

```
animation:myfirst 5s linear 2s infinate alternate;
```

注意：

- 简写属性里面不包含animation-play-state
- 暂停动画：animation-play-state:paused;经常和鼠标经过等其他配合使用
- 想要动画走出来，而不是直接跳回来：

```
animation-direction:alternate;
```

- 盒子动画结束后，停在结束位置：

```
animation-fill-mode:forwards
```

5.4. 速度曲线调节

```
animation-timing-function：规定动画的速度曲线，默认是"ease"
```

6.1. 3D转换

立体空间时由三个轴共同组成的

- x轴：右边是正值，左边是负值
- y轴：下面是正值，上面是负值
- z轴：往外是正值，往里是负值

最常用的是3D位移和3D旋转

主要知识点：

- 3D位移：translate3d(x,y,z)
- 3D旋转：rotate3d(x,y,z)
- 透视：perspective
- 3D呈现transform-style

6.2. 3D移动translate3d

语法：

```
transform:translateX(100px):仅仅是在X轴上移动
transform:translateY(100px):仅仅是在Y轴上移动
transform:translateY(100px):仅仅是在Z轴上移动(注意：translateZ一般用px单位)
transform:translate(x,y,z):其中，x,y,z分别指要移动的轴的方向的距离
```

6.3. 透视：perspective

如果想要在网页产生3D效果需要透视（理解成3D物体投影在2D平面上）

模拟人类的视觉位置，可认为安排一只眼睛去看

透视我们也称之为视距：视距就是人的眼睛到屏幕的距离

距离视觉点越近的在电脑平面成像越大，越远成像越小

透视的单位是像素

透视写在被观察元素的父盒子上面
d:就是视距，视距是一个距离人的眼睛到屏幕的距离
z:就是z轴，物体距离屏幕的距离，z轴越大（正值）我们看到的物体就越大

6.4. translateZ

6.5. 3D旋转：rotate3d

3d旋转可以让元素在三维平面内沿着X轴Y轴Z轴或者自定义轴进行旋转

语法：

```
transform:rotateX(45deg):沿着X轴正方向旋转45度
transform:rotateY(45deg):沿着Y轴正方向旋转45度
transform:rotateZ(45deg):沿着Z轴正方向旋转45度
transform:rotate3d(x,y,z,deg):沿着自定义轴旋转deg度（了解即可）
```

对于元素旋转的方向判断，我们需要先学习一个左手准则

左手准则：

- 左手的大手拇指指向X(Y)轴的正方向
- 其余手指的弯曲方向就是该元素沿着X(Y)轴旋转的正方向

transform:rotate3d(x,y,z,deg):沿着自定义轴旋转deg角度（了解即可）

x,y,z是表示旋转轴的矢量，是标示你是否希望沿着该轴旋转，最后一个标示旋转的角度
transform:rotate3d(1,0,0,45deg)就是沿着x轴旋转45deg
transform:rotate3d(1,1,0,45deg)就是沿着对角线旋转45deg

6.6. 3D呈现 transform-style

控制子元素是否开启三维立体环境

```
transform-style:flat子元素不开启3d立体空间，默认
transform-style:preserve-3d;子元素开启立体空间
```

代码写给父级，但影响的是子盒子
这个属性很重要，后面必用

7. 浏览器的私有前缀

浏览器私有前缀是为了兼容老版本的写法，比较新版本的浏览器无需添加

7.1. 私有前缀

- -moz-:代表Firefox的私有属性

- -ms-:代表IE浏览器的私有属性
- -webkit-:代表Safari、Chrome私有属性
- -o-:代表Opera的私有属性

7.2. 提倡的写法

```
-moz-border-radius:10px;
-webkit-border-radius:10px;
-o-border-radius:10px;
border-radius:10px;
```