Css笔记

1. transition过渡动画

过渡动画简单来说就是元素从一种样式变换为另一种样式

兼容性问题：

Internet Explorer 10、Firefox、Chrome 以及 Opera 支持 transition 属性。  
 Safari 需要前缀 -webkit-。  
 Internet Explorer 9 以及更早的版本，不支持 transition 属性。  
 Chrome 25 以及更早的版本，需要前缀 -webkit-。

1. transition-property属性

定义：对象中的参与过渡的属性名称

语法：transition-property：none | all | property;

说明：

none：没有属性改变

All：默认值，所有属性改变

Property：某元素的属性名

1. transition-duration属性

定义: 对象过渡的时间

语法：transition-duration：time；

例子：transition-duration：1s；

说明：

默认值0s

1. transition-timing-function属性

定义：设置对象中过渡的动画类型

语法：transition-timing-function：ease；

例子：transition-timing-function：linear

说明：  
 linear:线性过渡（匀速）  
 ease:平滑过渡（0--慢--快--慢），默认值  
 ease-in:慢--快  
 ease-out:快--慢  
 ease-in-out:慢--快--慢

1. transition-delay属性

定义：过渡效果何时开始时间

语法：transition-delay：time

例子 : transition-delay：1s  
 说明：指定秒或者毫秒数之前要等待切换效果的开始，默认是0

1. transition复合属性

定义：设置对象变换时的过渡

语法：transition ： property duration timing-function delay；

例子 : transition：all 1s ease 1s  
 注意：时间顺序不能乱

1. animation关键帧动画

animation简单说就是在一段固定的动画时间内暗中在某一频率内改变其CSS某个或某些值,从而达到视觉上的转换动画效果。

transition动画和关键帧动画的区别在于 transition动画需要有状态的改变触发动画。

1. animation-name属性

规定 @keyframes 动画的名称。就是@keyframes后面跟着的动画名称。

1. animation-duration属性

规定动画完成一个周期所花费的秒或毫秒。默认是 0。

1. animation-timing-function属性

规定动画的速度曲线。默认是 "ease"。

常见的动画速度参数：

linear：线性过渡。

ease：平滑过渡。

ease-in：由慢到快。

ease-out：由快到慢。

ease-in-out：由慢到快再到慢。

step-start：等同于 steps(1, start)

step-end：等同于 steps(1, end)

steps(<integer>[, [ start | end ] ]?)：接受两个参数的步进函数。第一个参数必须为正整数，指定函数的步数。第二个参数取值可以是start或end，指定每一步的值发生变化的时间点。第二个参数是可选的，默认值为end。

1. animation-delay属性

规定动画何时开始。默认是 0。也即是指动画延时执行时间。

1. animation-iteration-count属性

规定动画被播放的次数。默认是 1。

还有个无线循环关键字infinite,也即是反复循环播放动画。

1. animation-direction属性

规定动画是否在下一周期逆向地播放。默认是 "normal"。当然还有下列值：

reverse：反方向运行

alternate：动画先正常运行再反方向运行，并持续交替运行

alternate-reverse：动画先反运行再正方向运行，并持续交替运行

1. transform动画

transform 属性向元素应用 2D 或 3D 转换。

例子：transform:rotate(360deg) skew(-20deg) scale(3.0)translate(100px,0)；

\*表示常用的

transform值：

none：默认值，不转换

matrix(n,n,n,n,n,n)：定义2D转换，使用六个值得矩阵

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。

\*translate(x,y)：定义 2D 转换。

translate3d(x,y,z)：定义 3D 转换。

translateX(x)：定义转换，只是用 X 轴的值。

translateY(y)：定义转换，只是用 Y 轴的值。

translateZ(z)：定义 3D 转换，只是用 Z 轴的值。

\*scale(x,y)：定义 2D 缩放转换。

scale3d(x,y,z)：定义 3D 缩放转换。

scaleX(x)：通过设置 X 轴的值来定义缩放转换。

scaleY(y)：通过设置 Y 轴的值来定义缩放转换。

scaleZ(z)：通过设置 Z 轴的值来定义 3D 缩放转换。

\*rotate(angle)：定义 2D 旋转，在参数中规定角度。

rotate3d(x,y,z,angle)：定义 3D 旋转。

rotateX(angle)：定义沿着 X 轴的 3D 旋转。

rotateY(angle)：定义沿着 Y 轴的 3D 旋转。

rotateZ(angle)：定义沿着 Z 轴的 3D 旋转。

\*skew(x-angle,y-angle)：定义沿着 X 和 Y 轴的 2D 倾斜转换。

skewX(angle)：定义沿着 X 轴的 2D 倾斜转换。

skewY(angle)：定义沿着 Y 轴的 2D 倾斜转换。

perspective(n)：为 3D 转换元素定义透视视图。

1. Position

静态定位

Position：static；（默认）

相对定位

Position：relative；

绝对定位

Position：absolute；

固定定位

Position：fixed；

1. Flexbox

display：flex；

|  |  |  |
| --- | --- | --- |
| **属性** | **值** | **备注** |
| flex-direction | row（默认）/column | 指定主轴的方向 |
| flex-wrap | wrap/wrap-reverse | 换行 |
| align-items | stretch（默认）/center/flex-start/flex-end | 控制在交叉轴上的位置，align-self覆盖align-items |
| justify-content | flex-start（默认）/flex-end/center/space-around（均匀分布）/space-between（不会在两端留下任何空间） | 控制在主轴上的位置 |
| order | 数字，默认0，越大越后面 | flex 项排序 |
| flex | flex:0 1 auto（默认） | flex是三个属性的缩写：flex-grow/flex-shrink /flex-basis |

1. 浮动

清除浮动

1. 盒子上下左右居中
2. 网页布局

一列布局

margin：0 auto；

两列布局

三列布局

Scss笔记

SCSS 是 Sass 3 引入新的语法，其语法完全兼容 CSS3，并且继承了 Sass 的强大功能。

唯一不同的是，SCSS 需要使用分号和花括号而不是换行和缩进。

SCSS 对空白符号不敏感。

1. 在scss中 & 都代指父类

.fa{

&:hover{}

}

1. 变量

变量是以 $ 开头的，可以是颜色，长度，数值，等等。

像 js 的变量一样，scss 的变量也是有作用域的，也就是说内部声明的变量是无法在外面使用的，如果想让内部的变量在外部可访问，需要在变量值后面添加 !global 声明。

（还可以通过添加 !default 给变量设置默认值，这个在写一个样式库的时候比较实用，避免别人在没有给变量赋值之前使用）

$变量名：变量值

使用：

$redColor：red;

.fa{

Background-color：$redColor；

}

1. 嵌入字符串

将变量直接嵌入字符串，需要用 #{ 变量 }

#{} 可以放任意东西

使用：

$bg-path: "./img"

.fa{ background: url("#{$bg-path}/card-bg.png" center center); }

1. 导入文件@import

通过 @import 可以把多个文件结合到一起

使用：@import "文件名";

1. 相互嵌入@mixin @include

通过 @mixin 定义一个类或方法，在其它地方通过 @include 引用这个方法或类

使用：

@mixin a($cont：5px){ // 定义 mixin 并设置默认值 5px

border: $cont;

}

.fa{

@include a(10px); // 这里引用上面的 mixin，并设置值 10px

}

1. 继承@extend

有些时候，需要写的某个类里，包含另一个类的所有声明。

使用：

.color{

color：red；

}

.fa{

@extend .color; //可以继承多个用逗号隔开

}