CSS高级

一、兄弟选择器

匹配指定元素的相邻兄弟元素

1、相邻兄弟选择器

匹配相邻的元素；匹配当前元素【后面】的【一个】元素

前提是两者必须拥有相同父元素

\*注：只匹配后面紧挨着的那个元素，如果后面的元素不是，则不生效，不会继续往下匹配

语法：通过+作为结合符

例如div+p{}

2、通用兄弟选择器

匹配当前元素【后面】的【所有】【兄弟】元素

前提是必须拥有相同父元素

语法：通过~作为结合符

\*注：第二个元素不必紧挨着第一个元素后面

二、属性选择器

能够将元素附带的属性用于选择器中，对带有指定属性的元素设置样式

语法： 1、[属性名]，选择带有指定属性的元素

比如[title]，表示匹配包含title属性的所有元素

2、元素[属性名]

比如p[id]，表示匹配包含id属性的所有p元素

3、元素[属性1][属性2][...]，表示匹配即具备属性1又具备属性2的元素

4、元素[属性=值]

比如input[type=text]或者input[type="text"]，表示匹配所有type属性值为text的input元素

5、元素[属性~=值]，指定属性中，包含【独立值】单词的元素

比如 div[class="border"]，只能选择第一行，第二行不能识别

div[class~="border"]，则能两行都选

第三行都不能识别

<div class="border"></div>

<div class="border content important"></div>

<div class="myborder"></div>

6、元素[属性^=值]，属性值以【值】开始的元素

比如div[class^=c]，指定class属性以c开头的div元素

7、元素[属性\*=值]，属性值中包含【值】的元素

比如 div[class\*="border"]，除了前两行，第三行也可识别

<div class="border"></div>

<div class="border content important"></div>

<div class="myborder"></div>

8、元素[属性$=值]，属性值以【值】结尾的元素

比如 div[class$="er"]，第一、三行可识别，第二行不能识别，因为第二行是以t结尾

<div class="border"></div>

<div class="border content important"></div>

<div class="myborder"></div>

三、伪类选择器

1、目标（target）伪类

用于匹配当前活动的锚点元素

语法： ：target

2、元素状态伪类

场合：表单元素居多

:enabled 匹配每个已启用的元素

:disabled 匹配每个被禁用的元素

:checked 匹配每个已被选中的input元素（单选按钮、复选框适用）

3、结构伪类

（1）:first-child，匹配属于其父元素中的首个子元素

（2）:last-child，匹配属于其父元素中的最后一个子元素

（3）:empty，匹配没有子元素（包括文本）的元素

例如<p></p> 可以匹配出来

<p>

<a>百度</a>

<p> 不能匹配出来

<p>百度<p> 不能匹配出来

（4）:only-child，匹配属于其父元素中的唯一子元素

比如a:only-child{}能匹配出来

<p>

<a>百度</a>

<p>

a:only-child{}不能匹配出来

<p>

<a>百度</a>

<a>谷歌</a>

<p>

4、否定伪类

匹配非指定选择器的每个元素（即不是指定选择器的其他元素）

语法： :not(...)

例如 input:not([type=text])

tr:not(:first-child)

四、伪元素选择器

特点：获取指定元素中某一部分文本而用的

1、:first-letter，用于选取指定选择器（元素）的首字母

2、:first-line，用于选取指定选择器（元素）的首行文本

3、::selection，匹配被用户选取的部分

五、内容生成

通过CSS向已有的元素上增加新的文本或者图片内容

选择器： 1、:before，向匹配元素之前增加生成的内容，或定位到匹配元素开始的位置

2、:after，向匹配元素之后增加生成的内容

例如div:before{}

属性：1、content，配合:before或:after伪元素插入生成的内容

例如div:before{

content:"生成的内容...";

}

常用取值： （1）字符串：普通文本

（2）url：图像

（3）计数器

六、计数器

通过CSS定义一个计数器，在其他元素中可以使用该计数器生成的数字

1、属性：（1）counter-reset，用于定义计数器并设置初始值

初始值默认为0，可以设置为正、负、0，如有多个计数器，中间用【空格】隔开

例如 选择器{

counter-reset:计数器1 10 计数器2 20；

}

\*注：如果整个页面的任何一个元素都想使用到计数器，最好将其定义在body中；切记 如果将计数器定义在某个元素（body除外）当中，在使用时永远都是初始值

（2）counter-increment，设置每次使用计数器时的增量，默认值为1

属性值可以为正（递增）、负（递减）

\*注：哪个标签需要使用，就在哪个标签中声明

例如 div{

counter-increment:计数器名称 增减量值;

}

2、函数：counter(计数器名) 函数

作用：使用计数器创建出来的数字文本

七、多列

属性： 1、column-count，元素被分割的列数

2、column-gap，列与列之间的间隔大小，以px为单位

3、column-rule，列规则（定义列间隔符的宽度、颜色、样式）

例如column-rule:1px solid red;表示列与列之间的间隔符是1px红色实线

浏览器兼容问题： IE10和Opera支持多列

Opera偶尔不识别的情况下：-o-

Firefox：-moz-

Chrome和Safari：-webkit-

例如 div{

-moz-column-count:3;

-moz-column-gap:20px;

-moz-column-rule:2px solid red;

}

八、CSS Hack

1、CSS类内部hack（内部样式）

2、选择器hack

3、html头部引用hack

通过IE的条件注释来完成

条件注释：与html注释<!-- -->相似

if条件判断是否能够解析注释里的内容

语法：<!--[if条件]>满足条件要做的事情<![endif]-->

gt，选择条件版本以上的版本（不包含条件版本）

例如gt ie 7

lt，选择条件版本以下的版本（不包含条件版本）

gte，选择条件版本以上的版本（包含条件版本）

lte，选择条件版本以下的版本（包含条件版本）

！，选择条件版本以外的所有版本

例如 只在ie6中显示：

<!--[if ie 6]>这段文本只在ie6中显示<![endif]-->

九、转换

改变元素的一些状态，比如大小、位置、形状

可以是2D转换，也可以是3D转换

1、转换属性：transform

取值 none，不转换，默认值

transform-函数，表示要实现转换的函数

函数： rotate()，旋转

translate()，位移

scale()，缩放

2、转换原点

转换过程中围绕的中心点，默认原点是在整个元素的中间处

transform-origin，更改转换原点

取值： 数值，以元素左上方的点为（0,0）再去计算其他点

百分比，0%，0%表示元素左上方的点；默认值为50%，50%表示元素的中心点；100%，100%表示右下方的点

关键字，left、right、top、bottom、center，left top表示左上方

赋值： 一个值，表示x轴

两个值，表示x轴和y轴

三个值，表示x轴、y轴和z轴

3、2D转换--位移

从当前元素的位置处移动到指定坐标轴的位置处

函数：translate(x,y);

translate(value);

取值： 数值（可以取负值），x为正向右移；负向左移，y为正向下移；负向上移

百分比

单方向位移： translateX(x);横向

translateY(y);纵向

4、2D转换--缩放

改变元素大小（包括里面的内容，比如div框里面的文本）

函数： scale(x,y);

scale(value);

取值： 默认值为1（倍）

缩小：0-1之间的小数

放大：大于1的整数或小数

单向缩放： scaleX(x);横向

scaleY(y);纵向

5、2D转换--旋转

函数：rotate(...deg);

取值：deg，角度，0-360之间，默认为顺时针旋转；取值为负则为逆时针旋转

例如transform:rotate(45deg);表示顺时针旋转45°

6、2D转换--倾斜

函数：skew(...deg);

单方向倾斜skewX(x)、skewY(y)

7、3D转换

属性：perspective，设置假定人眼到投影平面的距离，取值为px，取值越大视觉效果越远，越小视觉效果越近，只影响3D元素不影响2D元素

设置位置：加在父元素上，设置好后，其子元素就可以完成3D转换

(1)3D位移

改变元素在z轴上的位置

属性：transform

函数： translateZ(z);

translate3d(x,y,z);

(2)3D旋转

函数： rotateX(...deg);

rotateY(...deg);

rotateZ(...deg);

rotate3d(x,y,z,...deg);xyz取值分别可为1 -1 0，1表示顺时针-1表示逆时针0表示不转

8、多个效果叠加

中间用【空格】隔开，比如transform:translate(100px;50px;) rotate(45deg);

十、过渡

元素从一个状态到另外一个状态的平滑变化过程

1、过渡属性

transition

2、过渡四要素（四子属性）

（1）过渡属性，即元素的哪个【状态】发生变化时要使用过渡的效果，当指定的属性发生改变时就会触发过渡效果

语法：transition-property

取值none、all、property（常用）

例如transition-property:background;

（2）过渡时间，即整个过渡效果在多长时间内完成

语法：transition-duration

取值以s（秒）、ms（毫秒）为单位，1s=1000ms

例如transition-duration:1s;

\*注：过渡时间默认为0，无过渡效果

（3）过渡函数，即指定时间内（transition-duration）过渡的【速度】效果

语法：transition-timing-function

取值： ease，默认值，慢速开始，逐渐变快，再慢速结束

linear，匀速开始到结束

ease-in，慢速开始，加速到结束（加速效果）

ease-out，快速开始，慢速结束（减速效果）

ease-in-out，以慢速开始和结束，先加速再减速

（4）过渡延迟，即当过渡操作被激发后等待多长时间后才开始执行过渡效果

语法：transition-delay

取值以s（秒）、ms（毫秒）为单位

3、统一设置（四个）过渡效果

transition:transition-property transition-duration transition-timing-function transition-delay;

4、触发过渡

过渡效果由用户行为进行触发（比如点击、鼠标悬停）

\*\*注：例如 #d1{

width:200px;

height:200px;

background:red;

}

#d1:hover{

background:blue;

}

过渡样式transition:background 1s linear 1s;

\*过渡样式放于#d1或者#d1:hover中都能实现过渡效果，但是如果将过渡样式放于#d1中，鼠标移开时则同样有过渡效果；将过渡样式放于#d1:hover中，鼠标移开时则没有过渡效果，瞬间恢复

5、多个过渡效果（同时执行）

中间用 , 隔开，比如 transition-property:property1,property2,...;

transition-duration:t1,t2...;

transition-timing-function:f1,f2...;

transition-delay:d1,d2...;

十一、动画

将元素的多个状态放在一起运行（多个状态（属性）之间的转换）

\*注：浏览器兼容性

Chrome和Safari: -webkit-

Firefox: -moz-

Opera: -o-

1、实现动画的步骤：

（1）声明动画， @keyframes

声明整个动画过程中的不同状态

（2）调用动画， animation

2、关键帧

语法： @keyframes

例如<style>

/\*兼容性\*/

@-webkit-keyframes 动画名称{

@keyframes 动画名称{

/\*定义关键帧，即不同时间点上的动画状态\*/

form或0%(表示时间，不一定为0开始，100以下均可){

/\*动画的开始状态（样式）\*/

background:red;

}

20%{

/\*动画在执行期间的状态（样式）\*/

...

}

...(若干关键帧)

to或100%(必须为100){

/\*动画的结束状态（样式）\*/

...

}

}

</style>

3、动画属性（调用）

(1)animation-name:调用动画的名称，指定 @keyframes的名称

(2)animation-duration:动画执行的时长，以s或ms为单位

(3)animation-timing-functioin:执行时的速度效果，取值ease、linear、ease-in、ease-out、ease-in-out

(4)animation-delay:延迟时间

(5)animation-iteration-count:动画执行的【次数】

取值： 具体数值

infinite，无限循环

(6)animation-direction:动画播放方向

取值： normal

alternate，奇数次播放为正向播放（状态从from-to）

偶数次播放为逆向播放（状态从to-from）

reverse，反向

(7)animation:动画综合属性（中间空格隔开）

animation:name(动画名) duration timing-function delay iteration-count direction;

(8)animation-fill-mode:指定动画在播放之前或之后的效果

取值： none，默认值，无效果

forwards，动画完成后，保持最后一个状态

backwards，反向运动的前提下，动画显示之前，保持在第一个状态

both，既有forwards的效果又有backwards的效果

(9)animation-play-state:定义动画播放状态（播放、暂停、停止），配合Js使用

取值： paused，暂停

running，播放

十二、CSS优化

1、作用： 减轻服务器压力

缩短服务器响应时间

提高用户的体验度

2、优化原则： 尽量减少http请求个数

页面顶部引入CSS

将CSS和JS放到外部文件中

3、CSS代码优化： 合并样式

缩小样式文件

选择更优的样式属性值

减少样式重写

【代码压缩】

不要在html中缩放图像

避免空的src和href