# CSS

CSS是一种层叠样式表，如果把HTML比作骨头，那CSS就是皮肤，只有骨头的页面是贼难看的。通常用CSS对HTML标签页进行样式修饰。他的难度就比HTML高的多，下面是关于CSS的知识点。

## 选择器

CSS提供了15种选择器，下面进行一一介绍：

元素选择器：p{}这个选择器是根据元素名来进行选择元素的。

**类选择器**：.my-class{}这个是根据元素的类名来选择元素的。

**id选择器**：#my-id{}这个是根据元素的id来选择元素的。

**属性选择器：**a[href=”dwaw”]，[class=”a”]这个是根据标签的属性进行元素选择，其还有很多选择细节



在括号中的最后加一个i可以大小写一视同仁。

**伪类选择器**:last-child，在一堆兄弟元素中选择最后一个兄弟，如果没有兄弟就不选。

:only-child,独生子女，在存在父类元素的情况下子类元素只有这一个。

:hover,鼠标经过元素时就会选中。

:focus，鼠标聚焦到元素时会选中，像是点击之后就会选中

:nth-child(2n+1),按一定规律选择子元素，是参照所有类型相同的兄弟选择。男女通吃。

:nth-of-type(n)，按照标签相同兄弟位置选择就像是选男不选女这样的与nth-child做一个区别。这些是比较常用的伪类选择器，剩下的在MDN随用随查即可。

**伪元素选择器**：

::before,相当与在后边加了一个东西，

::after,相当于在前边加了一个东西，对这个玩意要注意在这个玩意里面一定有content不然不会显示。即使是空的content也会有占位符的作用。

这俩是常用的，通常用于ico的添加之类的。剩下的在MDN用到查就行了。

**组合选择器：**

满足多个条件的选择器，比如div.myclass这种选择的就是有myclass类名的div

**后代选择器：**

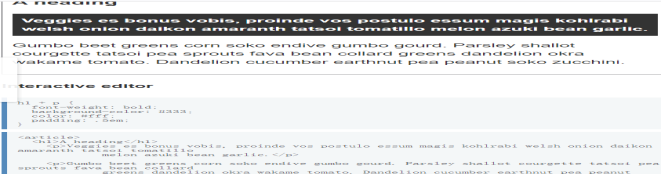
对具体爷爷的具体儿子，孙子进行选择，比如div p这样的。选择的就是div里的下一级p元素和下下一级的p元素

**子选择器：**

对父元素的所有儿子进行选择，比如div>p选择的就是div里的下一级p元素。

**相邻兄弟选择器：**

对相邻的元素进行选择比如说h1+p，这个就是对h1下一个兄弟p进行选择。



**通用兄弟选择器：**

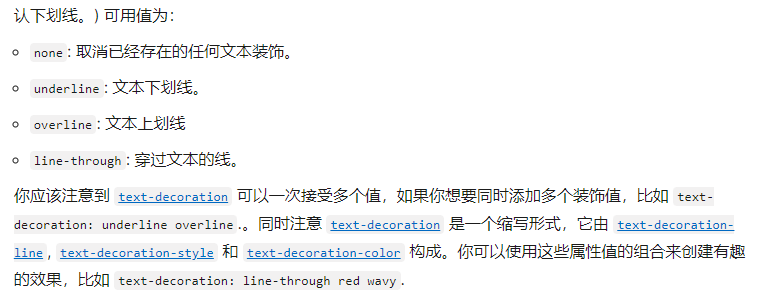
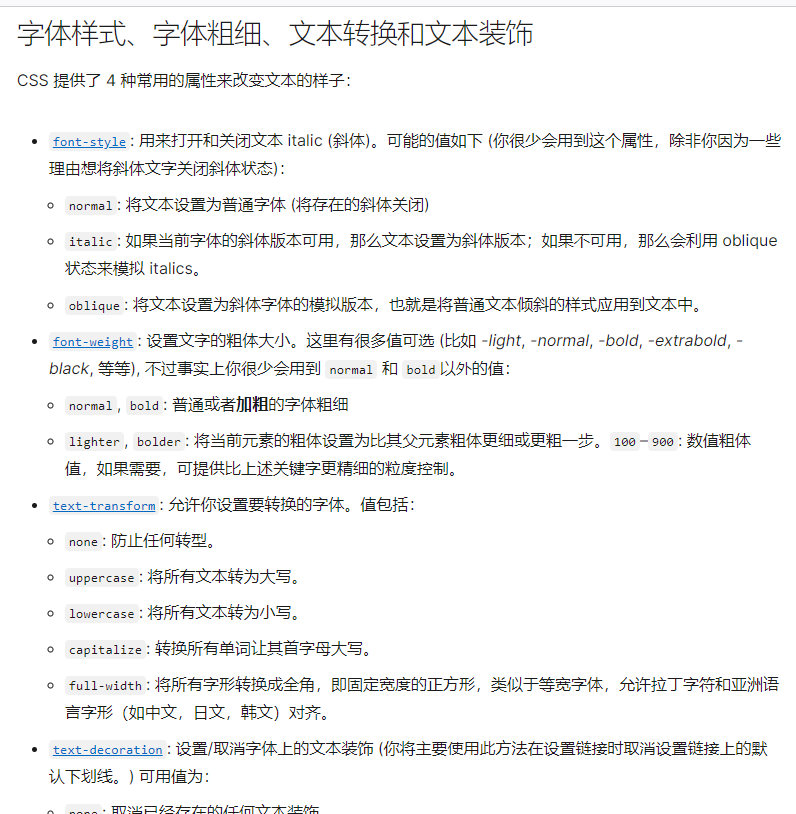
对所有兄弟元素生效，h1~p这种就是对与h1所有同级的兄弟都管用的选择器。

\*这个选择器选的就是整个body框子

## 文本属性

**字体：**font-family属性，调整元素的字体，具体字体在MDN可以看到。

**文字阴影：text-shadow**



**文本布局：**text-align:line-height:这种都是文本布局属性，还有text-indent这种文本缩进就不细说了，mdn上都有

**简写：**上述的所有文本属性其实都可以通过font：简写后面按一定顺序写就行了。

文本多行显示：

display: -webkit-box;（和 1 结合使用，将对象作为弹性伸缩盒子模型显示 ）

-webkit-line-clamp: 2;（用来限制在一个块元素显示的文本的行数，2 表示最多显示 2 行。为了实现该效果，它需要组合其他的 WebKit 属性）

-webkit-box-orient: vertical;（和 1 结合使用 ，设置或检索伸缩盒对象的子元素的排列方式 ）

overflow: hidden;（文本溢出限定的宽度就隐藏内容）

text-overflow: ellipsis;

white-space: nowrap控制是否换行。

## 背景

盒子要有具体的背景来进行一些装饰，不然太难看，这就是我们的background属性按照一定顺序就可以完成。但通常都是分开写的。

常用的有以下几个：

background-color:设置背景颜色。

background-image:设置背景图片。

background-size:设置背景图片的大小。

background-clip:设置背景图片的延伸范围，是否到达盒子边界这样的。

background-position:设置背景图片在盒子中的位置。

background-attachment:设置背景图片是否随着图片移动滚动。

## CSS属性计算机制

CSS想进行属性计算要先把选择器匹配了，然后再继承，再根据优先级确定，然后再把百分比转换为px，再进行计算。

## CSS视觉格式化模型

**盒子模型：**就是通过width，height，margin，padding控制盒子大小，等各种属性，和通过各种display来控制盒子的样式。是行内元素块元素，还是行内块元素。

**定位：**元素定位分为普通流，浮动，定位。普通流是从左到右，从上到下运动的。

浮动脱离普通流，位置向左，向右知道碰到边框结束，浮动的开始的初衷是做文字环绕效果，后来被用来做布局了。定位从数据流中清除了。位置由left,right,top,bottom来决定。

**Postion:**定位分为，绝对定位，固定定位，相对定位，粘性定位。

postion：static，相当于没有定位。没有脱离文档流。

postion:absolute，这个是根据上一个父类非static定位决定的。

postion：relative，这个直接就是根据浏览器视口定位的。其本质就是给绝对定位当爹。

Postion：fixed，固定定位，直接把元素位置定死在视口内了。

postion：sticky,粘性定位，在一定范围内是绝对定位，但是超过一定范围就成了固定定位。

**Float：**定义了元素浮动，float意味着启用块布局。

float：left;向左浮动

float：right;向右浮动

float：inline-start；向容器起始浮动；

float：inline-end;向容器结束浮动。

浮动的盒子会脱离文档流从而影响下面的盒子，盒子通常不定义高，靠内容撑起来，这种情况会导致塌陷这种情况需要清除浮动。

清除浮动有三种，一种是在后面加空盒子，一种是加伪类同时在IE中用\*zoom=1，还有一种给父元素加overflow：hidden不过无法溢出换行。不推荐用。

***层叠上下文：***z-index可以把顺序改成想要的，属于优先级问题

## 6.伪类和伪元素

在上面的选择器那部分就提到过，这里做一个细节说明，伪类是一个类似于class类的东西，可以帮助你选择元素。伪元素是类似于元素的东西，但是他又不是真的元素，他能起到元素的部分作用。

## flex和grid布局

**Flex布局：**

display:flex可以开启flex布局。

flex-direction:可以控制flex布局的方向，是row行还是column列。如果加上-reverse就会把起点反过来。

justify-content：元素不能填满，这个决定了盒子内元素的排版方式。

flex-start（默认值）：左对齐

flex-end：右对齐

center： 居中

space-between：两端对齐，项目之间的间隔都相等。

space-around：每个项目两侧的间隔相等。所以，项目之间的间隔比项目与边框的间隔大一倍。

align-items 属性描述：与justify-content功能相似但他是纵轴。

flex-start：交叉轴的起点对齐。

flex-end：交叉轴的终点对齐。

center：交叉轴的中点对齐。

baseline: 项目的第一行文字的基线对齐。

stretch（默认值）：如果项目未设置高度或设为auto，将占满整个容器的高度。

align-self:对应的是单个元素的align-items

**flex-wrap:负责换行的，不然堆一起难看呢**

Flex-shrink，flex-grow，flex-basis:这个是控制缩放的，flex-basis是缩放比例控制。

**Grid布局：**

与flex布局类似，但是grid布局更加牛逼他是一个二维布局。

display：grid创建网格布局。创建后不进行参数调整是不会有变化的，不会像flex一样变化。

grid-template-columns,grid-template-row：调整grid网格的行数和列数。

例：grid-template-row：100px,1fr,1fr。Fr是占剩余空间的百分比比例。grid-column-gap,grid-row-gap：这两个参数，是规定网格之间的距离的。可以合并成grid-gap；如果只有一个值就是行列共用间距，如果两个值就是先行后列。

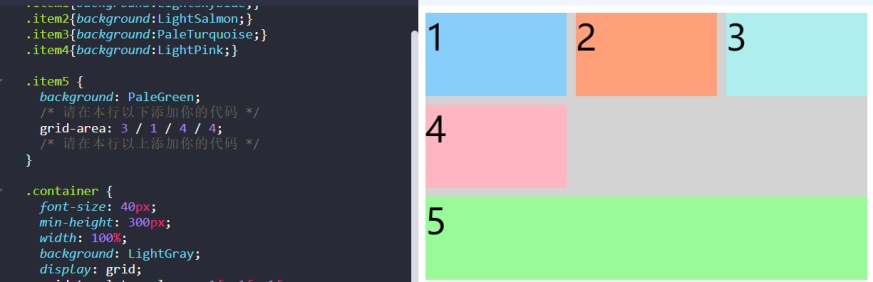
grid-row，grid-colum:可以规定网格项开始结束的列数。1/3就是从第一个槽到第三槽占两列。Row也一样的。

align-item,align-self,justify-self,justify-item;分为start，center，end，stretch决定了怎么填充单元格。

Grid-area，grid-template-area;前者是定义名字后者是调用名字。例子如下：



他还能网格位置



结合几个函数

repeat(3,1fr)=1fr,1fr,1fr，简化了代码。

minmax(100px,1fr)决定了一个范围。

Auto-fill，auto-fit是用来填充网格自适应的。

网格是可以嵌套的

place-items：相当于justify-items和align-items的一个结合。

## 媒体查询与响应式

响应式：根据不同页面大小进行自适应。

媒体查询：他就是一种响应式，通过不同设备视口大小来调整样式，例如：

@media(min-height=100px){}也就是设备高超过100px就会启用样式。

## 过渡transition

进行一个渐变效果，通常用于hover或者JavaScript的交互，例如这个：transition：margin-left 2s，他的意思就是margin-left变化有两秒的时间，他中间是以插值的形式表现出来的。系统会计算出插值并赋值。

## 渐变

liner-gridient，Radio-gridient等等，这种通常是为了设置颜色用的。

## 变形transform:

包括了平移，缩放，旋转

translate(x,y)平移，scale(n)缩放，rotate（deg）旋转，可理解为绕着Z轴旋转，和skewX()沿着x或者y轴转有一种立体感。

## 关键帧动画。

@keyframs name{

from{style}

to{style}

}，from，to也可以换成50%这样的百分比数字

P{

animation-name:name

animation-duriation：3s  
}

这就是一个例子，还有很多属性现用现查吧，animation还可以连写就不细说了。

他中间是以插值的形式表现出来的。系统会计算出插值并赋值。其本质就是你提供几个帧的效果，插值变成一个连续帧，实现动画效果。

## CSS预处理器（Less,Sass,Stylus）

预处理语言都很相似，这里就写一个Less。

Less作为预处理语言，可以让CSS更加接近编程语言，同时用LESS的嵌套等方法，也能提高大量CSS代码的可读性。Less编写时会自动生成一个CSS所以只管引入CSS就行了。

这部分可能不是很主流，

**Less变量：**

**选择器变量：**先声明@mychoose:#wrap,使用{@mychoose}{}这就相当于#wrap{}

**属性变量：**先声明@color:red;使用.wrap{color:@color}这就相当于.wrap{color:red}

**URL变量：**声明@img：../img,使用.wrap{background:url（“{@img}/rr.png”）}

相当于.wrap{background:url（“../img/rr.png”）}

**声明变量：**@baground:{CSS代码}使用时.wrap{@baground()}，相当于.warp{css代码}

**变量运算：**运算就是在值后面加减就行了不用加单位

**变量作用域：**与JS作用域差不多，都是就近

**用变量去定义变量：**@var：@color就是这种语法

**Less嵌套：**

对媒体查询@media很好用

@Media screen{

@media(min-height:100px)  
{width:100px}

}等价于@media screen and （min-height:100px）{width:100px}

&就相当于父元素比如说我们创建一个id伪father的父元素里面放ul放li

我们用嵌套设置属性可以这么写：

#father{

& ul{

li{}

}  
}这样可以提高代码可读性也可以提高使用感。

**Less的引入**

[导入@import{nav}等于@import{nav.less}，实际开发大概就用这些了把。](mailto:导入@import{nav}等于@import{nav.less}，实际开发大概就用这些了把。)

还有一些语法去文档看吧，不常用，用了现看就行。

**Less的混合**

比如说上面写了一个.mix(){css样式}，下面就能使用mix1(){.mix()}这样类似的。他也能接受传参。

总结：就是一句话，less做为一种CSS预处理语言，他牛逼之处就在于他能以少量的代码干更多的事，这就帮助了我们CSS解决了难维护的问题。提高了效率。毕竟编译阶段不算在运行时间里。