cascading style sheets 层叠样式表，用于做出给人看的HTML，这里仅记录最基础也最常用CSS属性，毕竟我只是搞JAVA的。

////////////////////////////////////////////////////////////////////////////////////////////

~ 选择器

1. 标签选择器

这是选择器->p{

选择哪个标签 font-size: 20px;//修改字号

color: red;//设置颜色

........

}

标记选择器的缺陷很明显，不能复用，这怎么符合我们面向对象的套路，因此，这里使用类别选择器。

2. 类别选择器

//这个名字-随你起

这个点不能漏->.r{

color: red;

font-size ： 10px;

....

}

接着在标签中这样写

<p class = "r">

3. ID选择器

#name{ //写属性 }

一次性的东西，不推荐用。

4. 交集选择器

交集复合选择器，由两个选择器组合而成，第一个是标签选择器，第二个是类选择器或ID选择器。既保留了标签选择器的样式，也保留了类别选择器的样式，同时还增加了自己的新样式。

p{

color: red;

}

.c{

font-size: 30px;

}

p.c{

text-indent: 2;

}

此时，<p class = "c"></p>内的字就具备了以上三种属性

5. 并集选择器

有多个选择器通过逗号连接在一起（标签选择器，类别选择器，ID选择器）

h1，c，#re{

backgroun-color：grey;

}

统一给使用这三个选择器的标签加背景色

6. 后代选择器

标签嵌套，一层一代。

<div>A<p>B<span>C</span></p></div>

这是一个三代。

<span>D</span>

这样写的话，C与D都会变成蓝色。但如果你只想让C变成蓝色呢。

span{

color: blue;

}

这样，只要被<p>嵌套的得<span>就会变成蓝色。另外 P 前面还可以接着写，写好多。

P span {

color: blue;

}

7. 子选择器

对后代选择器的完善，因为后代也分直接后代与间接后代（儿子与孙子）。有时候你只想给儿子特定样式，不希望孙子也有，这就要用到子选择器

div>span{

color: blue; //像这样

}

/////////////////////////////////////////////////////

CSS的继承特性

有些样式可以继承，如<ol>与<ul>中嵌套的<ol>或<ul>会继承上一个列表的样式，但也不是所有样式都会继承。

/////////////////////////////////////////////////////

好了，现在问题来了，CSS要怎么与ＨＴＭＬ协同呢？

～行内样式

　　＜ｐ style = "color ：red;font-size：10px;"＞

这玩意看着挺像变量声明的。问题也挺明显，首先不是什么地方都有style属性的，而且就算可以写也不能复用。

~内嵌式

上面写的那些个标签就是内嵌式

~链接式

你写一个 .css 文件，（这样挺好的，统一管理）

然后将该文件的样式连接到HTML文件。这个时候就要在<head>里写个标签

<link href = "" type = "text/css"

rel = "stylesheet">

有点像<a href = ""/>就是多了两个属性，这三样都是必须的，缺一不可。href没什么好讲的，超文本引用。type，类型。rel, relationship的缩写，必须的，就这么规定的。

~优先级比较

自行百度，反正行内样式第一，平时多用链接式就是了。

/////////////////////////////////////////////////////

~常用CSS属性

属性后写冒号：不是 = ！！

color: //前景

background-color: //后景

text-align：//文本对齐

padding：//边距设置

width：//宽度

height：//高度

float：//图片悬浮，文字围绕

margin：//增大DOM节点间距

font-size：//字体大小，单位px

font-family：//如果浏览器不支持第一个字体，那就换下一个。

line-height：//行高，1.5最好看

text-indent：//缩进，2em缩进两个字

~对页面进行优化

body{

margin：0px；

background-color：yellow；

}//不过颜色这玩意最好通过16进制数字来设置，可选择范围大。

/////////////////////////////////////////////////////

~文本样式

//这是个英文字体

font-family："Times New Roman",...；

//第一种没用用第二个，依次类推

font-weight：bold;

//文字加粗

font-weight：normal；//不加粗，开关。

text-transform：capitalize；

//英文字母大小写转换

text-transform：uppercase；

//全大写

text-transform：lowercase；

//全小写

font-size：36px；

//文字大小，单位居然是像素px

font-decoration：none；//这个是没有

//文字的装饰效果，下划线之累的

font-decoration：underline；//下划线

font-decoration：line-through；//删除线

font-decoration：overline；//上划线

text-indent：2em；

//段落首行缩进

word-spacing：10px；//单词

letter-spacing：2px；//字母

//字词间距，一个中文字就算一个字母，就这么规定的。（规定的莫名其妙）

line-height：1.5；//这个是最漂亮的。默认1.2

//行高

margin：0px；

//DOM节点间的边距（<p>与<p>之间）

padding：2px；

//内部放大，一个是外边距，一个是内边距，单位都是px。

border : 1px；边框。

//齐了，一个节点就包含这三部分

text-align：center；

text-align：left；

text-align：right；

//文本对齐

text-align：justify； //向左向右对齐，这个还真不太好描述。它和居中（center）是不一样的。

段落垂直对齐，直接百度。

/////////////////////////////////////////////////////

~图片相关

1.边框

border-width：1px；//单位为px

border-color：red；

border-style：solid；//实线边框

border：2px red dashed；

//连在一起写，dashed是虚线

dotted//纯点组成的线

//给不同的边框设置样式

border-top：2px red solid；

border-right：2px blue dashed；

border-left：......

2.大小

//图片缩放

width：848px；height：480px；

//按比例缩放

width：20% //这个百分之多少是根据容器的大小来设置的（div）

3.图文混排

不设置效果的话就是图片与文字上下分开显示。

float：left；//文字向左环绕还有right

margin：1px；//设置外边距，不设的话整个贴在一起，比较难看。

4.对齐

vertical-align：baseline；基准线对齐。相应的还有一大堆，我靠烦的要死。

w3school上有，多查文档。

/////////////////////////////////////////////////////

~背景

//背景颜色

1.background-color：.....；//这总不用讲了吧

//背景图像

2.background-image：url(....)；//默认效果是平铺（全铺，上下左右），有时候好，有时候不好吧。

background-repeat：repeat-x；//横向平铺

repeat-y；//竖向

no-repeat；//不铺

background-position：20px 40px；//默认是放在左上角的，不太好看

background-attachment：fixed；//图片位置固定，不跟随滚动条走动。

/////////////////////////////////////////////////////

盒模型

从里到外是 padding border margin 他们都有top bottom left right 这几个元素。上面讲过一个border，至于padding与margin你，这两样看不见，只有border可以有实线。虚线。点线这种样式。

。。。。。。。。。。。。。。。。。。。。。。。

~块级元素与行内元素

<div>为块级元素，竖直排列，即padding自动撑满

<span>行内元素，横向排列，排满自动换行。

1.行内元素与行内元素之间的水平margin为left＋right

<span style = “margin-left： 10px;”></span>

<span

style = “margin-right： 10px;”></span>

那么两个<span>之间的距离为20px

2.块级元素之间的竖直margin，不是单纯地相加。

<div style = "margin-bottom: 10px;"></div>

<div style = "margin-top: 10px; "></div>

但是，这两个块级元素之间的距离任为10px。规则是这样的，距离以大的为标准，其中一个div变为20px，竖直距离才会变大。小的被大的吃掉了。

有意思的是，margin可以设成负的，那样就会重叠。

/////////////////////////////////////////////////////

盒子的浮动与定位

~盒子浮动

默认为none，按照标准流来定义。不悬浮，从头到脚排列。

float：left；//左悬浮，该盒子靠左，边距什么的也变小了

float：right；//右悬浮，该盒子靠右，边距什么的也变小了

~清除浮动

默认none，主要针对那些没设置float属性的盒子，不随着它们改变自身

clear：left；//不允许左边有悬浮

clear：right; //不允许右边有悬浮，但是，盒子高度各不同，所以有时候仅设置右悬浮或左悬浮，结果却是左右都不悬浮。

clear：both；//不允许有悬浮

~定位

默认static，反正不写就是默认。

//相对定位，以原位置为基准进行偏移。

position：relative；

left：30px； //以原位置为基准，距离左30px，

top：30px； 距离上30px

//绝对定位，以包含框为基准进行偏移，所以重点在与，包含框是哪一个。这个包含框的定义还是比较操蛋的，不看嵌套，要看最近的有position属性的盒子，static不行，默认等于没写。

这边牵扯出拎一个比较操蛋的玩意，那就是包含框的position属性值是哪个。

position：absolute；

right：30px；

bottom：30px；

//固定定位，就以浏览器窗口为基准进行定位。管你有没有框呢。

position：fixed；

/////////////////////////////////////////////////////

~z-index空间位置

比如说让文字浮在图片上方，又不想让图片成为背景的时候

z-index：-1；

/////////////////////////////////////////////////////

~display

display：inline；// div横着排

display：block；//span竖着排

/////////////////////////////////////////////////////

~表格样式

边框

1.caption标题、thead表头、tbody表格体、tfoot表尾。

2. .c td{ }

//这个写法。。。。在下孤陋寡闻了。使用该类的表格下的td标签的样式，需单独设置。

3. border-collapse：边框分离与合并，值为 separate ， collapse

4.border-spacing：单元格间距，这东西我觉得用的比较多。

5.padding，单元格内容与边框之间的距离，这属性，让我想起了cellpadding与cellspacing。但要多用CSS。

、、、、、、、、、、、、、、、、、、、、、、、

表格布局

默认为auto，宽度随内容改变

fixed 固定宽度，根据定义的width，不变。

、、、、、、、、、、、、、、、、、、、、、、、、、隔行换色

偶数行与奇数行的颜色不一样，在JSP中动态实现即可。

、、、、、、、、、、、、、、、、、、、、、、、、、临界选择器

th+td+td{ }

、、、、、、、、、、、、、、、、、、、、、、、、、鼠标经过时行变色

tr: hover{

background-color: aqua;

}

///////////////////////////////////////////////////////

~列表样式

<ul> ， <ol>。每一列要以<li>开头。无序列表默认实心圆，有序列表为数字。

1.list-style-type：square；

//这样就变成方形，有趣的是，该属性<ul><ol>通用。当然也可以单独设置。

2.list-style-image：url（“....”）；

图片符号，所以相应的，图片尺寸为16\*16px，大了也显示不出来。

///////////////////////////////////////////////////////

~CSS布局

一般分为HEAD，CONTENT，SIDE , FOOT。

所谓固定宽度，width = 1366px; 行了，固定死了。一般来讲，div会这么设

margin：20px 20px auto auto；

padding：20px 0px 20px 0px；

就是要留白，艺术上最重要的一笔。

1.绝对定位法

position，搞各种px，加上二分查找，有时候累，有时候爽吧。

float：left/right。

3.变宽度布局

用px就是固定布局，百分比就是变宽布局。随浏览器变化而变化，比较好看。但也不会无限制的缩小。

////////////////////////////////////////////////////////////////////////////////////////////

* margin-right为何无效

margin-right其实有效果的，只是在默认即标准流的情况的下显示不出来效果。如果脱离标准流呢？想到这个，就立马在css文件中加了一个：float:right;然后在测试的时候就能看到margin-right的效果了

关于right属性无效的原因：right属性只有在position是absolute的情况下才有效，而默认的position值是static，right属性是无效的。建议能不使用right就不要使用right属性。

* 写在<head>标签里的css类引用时没有效果

肯定是有效果的，因为规则就是这么设置的！没有出现想要的结果多半是因为冲突，比如已经引用了一个css文件，该文件已经对元素做了修饰，根据优先级（？），自己手写的css类便没有效果。而一旦放在style=””里（优先级最高），效果立马出来了

* width的 % 与 px

很奇怪的错误，我不能把div横向排列，就算横过来了，样式，超链接缺都失效了。

给div设置width为20%时，实际上是占满整个宽度只显示20%