**CSS（Cascading Style Sheets 层叠样式表）概述**

**为什么会出现CSS？**

因为HTML中不同的标签想完成相同的效果使用的是不同的属性，导致了样式不通用的问题，违反了单一责任原则，给维护人员带来了很大的麻烦。这时候就迫切需要能够一统样式江山的语言，随之，CSS就诞生了，CSS的作用是以统一的方式实现样式的定义，提高页面样式的可重用性和可维护性，实现了内容(HTML)与样式(CSS)的分离,遵守了单一责任原则,维护人员的春天也就随之而来。

**何为CSS？**

CSS全称是Cascading Style Sheets,中文名称是层叠样式表或级联样式表，简称样式表。

* Cascading 层叠: 当一个web页面使用多个样式表时，多个样式表中的样式可层叠为一个。
* Style 样式: 对HTML标签美化的属性。
* Sheets 表： 即把所有的样式集中存储起来，就形成一张表。

**HTML与CSS的关系？**

HTML构建网页的结构，CSS构建HTML标签的样式。

**HTML属性与CSS样式的使用原则:**

W3C建议尽量使用CSS样式取代HTML属性，但是一些HTML标签中无法替代的属性只能使用HTML元素的属性，不能使用CSS属性。

**HTML页面中有三种使用CSS样式表的方式：**

* 内联样式: 样式定义在单个HTML标签中，写在标签的style属性里，只能在单个标签中写css属性。
* 内部样式表: 样式定义在HTML页面head标签中的style标签中，可保存页面中所有HTML标签定义的css属性。
* 外联样式表: 样式定义在外部的一个后缀名为.css的文件中，通过link标签导入到HTML页面的head标签中，可保存页面中所有HTML标签定义的css属性。

内联样式表缺点很明显，违背了单一责任原则，使HTML页面看起来很乱，可读性很差;而内部样式表也违背了单一责任原则;只有外联样式表即遵守了单一责任原则，又使HTML页面很整洁，可读性较强。故强烈建议使用外联样式表导入css文件。

**个人学习CSS过程中的总结一下CSS特别之处**

**CSS样式表的层叠性和优先级：**

CSS使用层叠的原则来考虑继承、层叠次序和优先级等重要特征，从而判断相互冲突的规则中哪个规则应该起作用。

* 继承性是指：子元素可以直接使用父级元素定义的样式，大部分的css样式属性是可以被继承的，除了一些特别的标签需要重定义样式。
* 层叠性是指：当一个页面使用多个样式表时，若多个样式表中给同一选择器定义的样式没有相同的部分，则多个样式规则会层叠为一个规则，浏览器会显示所有的样式;反之，则通过优先级来判断使用哪个样式表中的属性。
* 优先级是指：当样式发生冲突时，浏览器首先会按照不同样式规则的优先级来应用样式。CSS样式的优先级如下所示（数字越大，优先级越高）：
  1. 浏览器缺省设置(User Agent StyleSheet)
  2. 外部样式表或者内部样式表,就近原则，谁在head标签中位置越靠后，最终就使用谁
  3. 内联样式
* !important规则：强制调整样式的优先级，无论该样式存放在哪里，都把该样式优先级提升到最高级别，通过(属性名称:值 !important;)使用，但是IE6及以下的浏览器不支持该规则，建议少用，因为有的时候在你不确定的情况下可能会打乱页面的布局结构。

**CSS中的框模型:**

在CSS中，每一个元素都被视为一个框，而每个框都有三个属性:

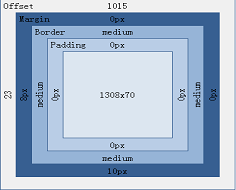
* border: 元素的边框(可能看不见)，用于将框的边缘与其它框分开;
* margin: 外边距，表示框的边缘距离相邻框之间的距离，也称为页边空白;
* padding: 内边距，表示边框与框内容直接的距离。

一个元素的实际宽度 = 左右外边距 + 左右边框 + 左右内边距 + width;

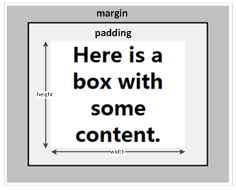
一个元素的实际高度 = 上下外边距 + 上下边框 + 上下内边距 + width;

在CSS中，有两种框模型，分别是IE盒子模型和标准W3C盒子模型。

1. IE盒子模型：IE的content部分包含了border和padding。如下图所示：



1. 标准W3C盒子模型：content、padding、margin、border，如下图所示：



由图可看出，width和height指的是内容区域的宽度和高度。增加内边距、外边距和边框不会影响内容区域的大小，但是会增大元素框的总大小。

因此，如果在一个具有边框的元素中放置文本，往往需要设置一下内边距，以便文本的边缘不要接触边框，这样便于阅读。而外边距则可以在多个元素框直接创建空白，避免这些框都挤在一起。因此，在设计页面是，经常会使用padding和margin属性来撑开页面。但是，必须注意的是，一旦用了padding和margin属性设置了元素的边距之后，会增加元素在页面布局中所占的面积。

**边框阴影特殊的地方**

属性：box-shadow

取值：

* h-shadow:必填的值，阴影的水平偏移距离，正值为右偏移，负值为左偏移；
* v-shadow:必填的值，阴影的垂直偏移距离，正值为下偏移，负值为上偏移；
* blur:阴影的模糊程度;
* spread:阴影的尺寸;
* color:阴影的颜色(默认是黑色#000);
* inset:将默认的外阴影设置为内阴影(若有特殊需求，可使用该属性（例如：通过行内元素实现按钮点击效果），一般不使用)。

简写 -- box-shadow: h-shadow v-shadow blur spread color inset;

有一个需要注意的地方 -- 那就是box-shadow属性可通过改变不同的取值等批量去写多个值，例如 -- box-shadow: h-shadow v-shadow blur spread color inset, h-shadow1 v-shadow1 blur1 spread1 color1, h-shadow2 v-shadow2 blur2 spread2 color2。

**文本阴影 -- text-shadow: h-shadow v-shadow blur color; 具体取值和需要注意的地方同边框阴影一致。**

**外边距和内边距特殊处理**

1. 外边距合并：

前提：只有普通文档流中的块框的垂直外边距相遇时才会发生外边距合并，行内块、浮动框或绝对定位之间的外边距是不会发生合并的。

当两个垂直的外边距相遇时，它们将合并成一个外边距，合并后的外边距高度等于两者之间高度较大者（谁高谁说了算），水平外边距永远都不会发生重叠。

1. 外边距溢出：在某些特殊情况下，为子元素设置上下外边距时，有可能会作用到父元素上;
   * 特殊情况：

1、父元素没有上或下边框;

2、必须为第一个子元素或最后一个子元素设置外边距时。

* + 解决方案：

1、为父元素增加边框（上或下）;

2、在父元素首行中增加空子元素<table>到第一个子元素（父元素:before{content:''; display:table;}）;

3、通过设置父元素的上内边距来取代子元素的上外边距。

1. 为行内元素和行内块元素设置外边距：

为行内元素设置上下边距无效，为行内块元素设置上下外边距会作用到整行内容都跟着变。

对行内元素和行内块元素设置上下内边距时，只会影响自己，并不会影响其它元素。

**box-sizing**

作用：

* 重新指定元素框模型的计算模式，元素边框内占地区域 = 左右边框+左右内边距+width。

取值：

* content-box -- 采用默认计算模式计算元素的实际尺寸。
* border-box -- 元素的尺寸会包含边框和内边距。

**为什么建议设置背景图像的同时还设置背景颜色？**

一般建议在使用背景图像的同时提供background-color属性，并且将其设置为接近背景图像颜色的颜色，可用吸色器获取背景图像的颜色。这样，如果正在加载页面，有的时候可能因为各种原因而无法显示背景图像时，页面可以使用这种颜色作为背景色，从而不会使页面布局变的很丑陋，使用户厌恶。

**在设置文本的字体时，为什么建议设置替换字体？**

因为如果我们设置的字体，用户电脑上没有安装指定的字体，则会使用默认字体。因此，如果可以指定一种替代字体，替代字体和指定字体不完全相同，相似且不会影响页面的布局，则可以解决这一问题。我们可以为font-family属性指定多种字体，且多种字体之间使用逗号隔开，则浏览器会通过查找字体列表中的下一种字体作为替代字体，如果找遍了字体列表还是没有可以使用的字体，浏览器才会使用默认字体。

所以，建议在所有font-family属性中提供一个通用字体序列。这样就提供了一条后路，在用户机无法提供与规则匹配的特定字体时，就可以选择一个通用字体作为替换。

**什么情况下需要额外设置表格的显示规则？**

1. 默认情况下(不额外设置表格的显示规则时候)，表格按照自动表格布局进行显示，即浏览器在显示表之前查看每一个单元格，然后基于所有单元格的设置计算表的大小，而列的宽度是由列单元格中没有换行的最宽的内容设定的。此时，单元格的大小会适应内容的大小，自动表格布局的算法在表格复杂是会比较慢，这是由于它需要在确定最终的布局之前访问表格中所有的内容。在不能提前确定每一列的大小时，这种方式非常适用。
2. 如果不想使用默认的自动表格布局，则可使用固定表格布局，设置table-layout属性的值为fixed。在固定表格布局中，水平布局仅取决于表格宽度、列宽度、表格边框宽度、单元格间距，而与单元格的内容无关。浏览器将适用某列指定的宽度来计算布局(如果给了宽度的话)，并使用该宽度计算该列中所有其它单元格的宽度。

自动表格布局 VS 固定表格布局

1. 自动表格布局
   * 单元格大小会适应内容大小;
   * 缺点：表格复杂时，加载速度慢;
   * 适用于不确定每列大小的情况下使用;
   * 虽然算法较慢，但是能体现传统表格的特点。
2. 固定表格布局
   * 单元格大小取决于td中设置的值;
   * 优点：加载速度快;
   * 适用于确定每列大小的情况下使用;
   * 算法较快，缺点是不够灵活。

**display:none; VS visibility:hidden; 和 opacity:0;**

display:none; 真正隐藏元素，脱离文档流，不占据页面空间，没有宽高，不会影响页面布局。

visibility:hidden;和opacity:0; 只是视觉上隐藏元素，但并不脱离文档流，导致空间依然被占据，保留自己的宽高，可能会影响页面的布局。

**CSS 中的定位机制**

CSS中，除了默认的流定位方式以外，还有以下几种定位机制：浮动定位、相对定位、绝对定位和固定定位。

流式定位(position:static;)是指元素框正常生成，块级元素生成一个矩形框，作为普通或文档流的一部分，行内元素则会创建一个或多个行框，置于其父元素中;

浮动定位(float:left/right;)是指将元素排除在普通流之外，并且将它放置在包含框的左边或者右边，但是依旧位于包含框之内，而且不能实现元素的堆叠，任何元素都可设置浮动定位，一经设置即变成块级元素，无论它之前是何种元素。

如果需要实现元素的灵活定位，且能够实现元素的堆叠，则可使用其它三种定位方式。其中：

1. 相对定位(position:relative;)将元素相对于它在普通流中的位置进行定位,元素仍保持其未定位前的形状，它原本所占的空间仍保留;相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置;
2. 绝对定位(position:absolute;)是指将元素的内容从普通流中完全移除，并相对于其包含块定位。包含块可能是文档中的另一个元素或者是初始包含块(一般为父元素),元素原先在正常文档流中所占的空间会关闭，就好像元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框，并且可以使用偏移属性来(top/right/bottom/left)固定该元素的位置;
3. 固定定位(position:fixed;)是指将元素的内容固定在页面的某个位置,类似与绝对定位。

补充一点：z-index属性，作用是在元素出现层叠效果时，改变它们显示的顺序，取值为数字，值越大，越靠上;只有在进行定位的元素中使用才有效，在其它元素中使用一律均无效。

**浮动引发的特殊效果**

1. 当父元素的宽度显示不下所有的已浮动元素时，最后一个将换行（当倒数第二个浮动元素高度大于最后一个浮动元素高度时，被会被卡住）;
2. 元素一旦浮动起来之后，那么宽度将变成自适应（非人为指定情况下）;
3. 元素一旦浮动起来之后，那么将变成块级元素，尤其对行内元素影响最大（可修改宽高度）;
4. 文本、行内元素、行内块元素是采用环绕的方式排列的，是不会被浮动元素压在低下的，会巧妙的避开浮动元素。

**浮动元素对父元素带来的影响**

由于浮动元素会脱离文档流，所以会导致元素不占据父元素的页面空间，所以会对父元素高度带来影响。如果一个元素中所有的子元素全部都浮动了，那么该元素的高度就是0.解决方案如下：

1. 直接设置父元素的高度，有个弊端：必须要知道父元素的精确高度，不好设置;
2. 设置父元素也浮动，弊端：会对后续元素有影响;
3. 为父元素设置overflow，取值为hidden或auto，弊端：如果有子集内容要溢出显示的话，也一同被隐藏。
4. 为父元素末尾追加空子块级元素，并设置clear属性为both（父元素:after{content:''; display:block/table; clear:both;}）。

**li列表项特殊的地方**

1. 列表项标识 -- 属性：list-style-type

取值：

* + none：无标记
  + disc：实心圆
  + circle：空心圆
  + square：实心方块

1. 列表项图像 -- 属性：list-style-image
   * 作用：有自定义图像作为显示的标识;
   * 取值：url(图像路径)。
2. 列表项位置 -- 属性：list-style-position

取值：

* + out-side:默认值，列表项标识位置在列表项之外；
  + inside:将列表项标识放在列表项区域之内。

1. 简写 -- list-style: type url() position;

常用方式为 -- list-type:none;

1. 列表项使用场合 -- 列表项能够使用在 纵向排列 和 横向排列 的元素中。

**content元素的作用:**

content元素与 :before 及 :after 伪元素配合使用，来插入生成内容，可以在元素之前或之后放置生成的内容。可以插入文本、图像、引导，并可以结合计数器为页面元素插入编号(一般在文章类项目使用)。

例如 -- 为匹配当前第一个子元素之后的所有同级指定元素内容区域之前插入文本：

父元素>li+li:before{content:'文本'}

**在a标签上设定伪类的顺序**

* a:link -- 普通的、违被访问过的链接
* a:visited -- 被访问过的链接
* a:hover -- 鼠标指针移动到链接上方
* a:active -- 鼠标点击链接的时候

为什么会规定这个顺序？

因为这是能够保证这四个所有伪类都被触发的方式，我们都知道，程序的代码是从上到下、从左到右执行的，最容易触发的放在最后面，则不会影响其他伪类被覆盖调;这同时也给我们一个启发，当我们要实现同一个元素但是要具有多种功能时，一定要注意层级关系，搞清楚功能之间有没有耦合。

**伪类选择器 -- 作用：匹配元素的不同状态，匹配得到的是元素。**

1. 目标为类：
   * 作用 -- 突出显示活动的HTML锚元素
   * 语法 -- elem:target;
2. 元素状态伪类：
   * 作用 -- 匹配元素的状态，主要使用在表单控件上；
   * 语法：
     + elem:enabled; 匹配已启用的元素；
     + elem:disabled; 匹配被禁用的元素；
     + elem:checked; 匹配被选中的元素（单选、多选按钮），
3. 结构伪类：
   * 作用 -- 通过元素结构之间的顺序，来匹配元素；
   * 语法:
     + elem:first-child 匹配属于其父元素的第一个子元素；
     + elem:last-child 匹配属于其父元素的最后一个子元素；
     + elem:nth-child(n) 匹配属于其父元素中的第n个子元素；
     + elem:empty 匹配没有子元素的元素；
     + elem:only-child 匹配只有一个子元素的父元素.
4. 否定伪类：
   * 作用 -- 匹配除了满足条件的选择器所匹配的元素，即获取剩下所有不满足条件的选择器所匹配的元素；
   * 语法 -- elem:not(选择器);

**伪元素选择器 -- 作用：匹配元素内的某部分内容，匹配得到的是元素的内容。**

1. elem:first-letter; 或 elem::first-letter; 匹配元素的首字符；
2. elem:first-line; 或 elem::first-line; 匹配元素的首行；
3. elem::selection; 匹配被选取的部分内容；

: 与 :: 的区别

1. : 既能表示伪类选择器，又能表示伪元素选择器(CSS2);
2. :: 在CSS3中表示伪元素选择器;

总结 -- 尽量选中 : 来使用。

**2D转换需要注意的地方：**

transform:rotate(ndeg);

* n为正值时，顺时针旋转；n为负值时，逆时针旋转；
* 旋转的时候，需要注意一下转换原点的位置，默认是在元素的中心点；旋转是围绕着转换原点来做的，所以原点位置不同，转换的效果可能与期望的有差异；
* 旋转的时候连同坐标轴也一同跟着旋转。

transform:skewX(ndeg);

向横向产生倾斜效果，实际改变的是y轴的倾斜角度；n为正值时，y轴按逆时针倾斜；n为负值时，y轴按顺时针倾斜。换成Y轴的话，纵向旋转x轴，原理同上。

**3D转换需要注意的地方:**

perspective属性，作用 -- 俗称视镜，模拟人眼到3D变换物体之间的距离，该属性要添加在3D转换元素的父元素上。

transform:rotate3D(x,y,z,ndeg);

* 当x、y、z的值设置为0时，说明该轴不参加旋转；
* 当x、y、z的值设置为1时，说明该轴参加旋转；

transform-style属性，作用 -- 如何处理子元素的摆放位置；

取值：

* flat,默认值，不保留子元素的3D位置；
* perspective-3d,保留子元素的3D位置，定义在要设置3D旋转的元素上。

**CSS中一些有意思的地方**

* visibility:collapse; 表格标签专用，效果等同display:none; 其它标签使用都等同visibility:hidden; 目前chrome不支持，FF、Opera、IE11支持。
* 光标cursor属性:

**解决CSS中一些属性兼容性问题的方法**

1. CSS Hack
   * 什么是CSS Hack？在head中引用Hack，使用IE条件注释来解决兼容性问题。
   * 语法： <--[if 条件 IE 版本]> 要执行的操作 <[endif]-->
     1. 版本 -- 取值：
        + 6~10直接的任意数字；
        + 省略版本，则判断是否为IE浏览器。
     2. 条件 -- 取值：
        + gt(greater than) -- 只在大于指定版本的浏览器中解析内容；
        + lt(less than) -- 只在小于指定版本的浏览器中解析内容；
        + gte(greater than or equal) -- 只在大于或等于指定版本的浏览器中解析内容；
        + lte(less than or equal) -- 只在小于或等于指定版本的浏览器中解析内容；
        + ! -- 只在条件版本以为的所有版本浏览器中解析内容；
        + 省略 -- 判断是否为IE 或 是否为指定版本。
2. 为属性或值添加浏览器前缀，特别是CSS3中的属性或值，建议一定要添加浏览器前缀。