background: linear-gradient(135deg, #FFD6E7, #FFC1C8, #FFD2C8, #FFB5E6);

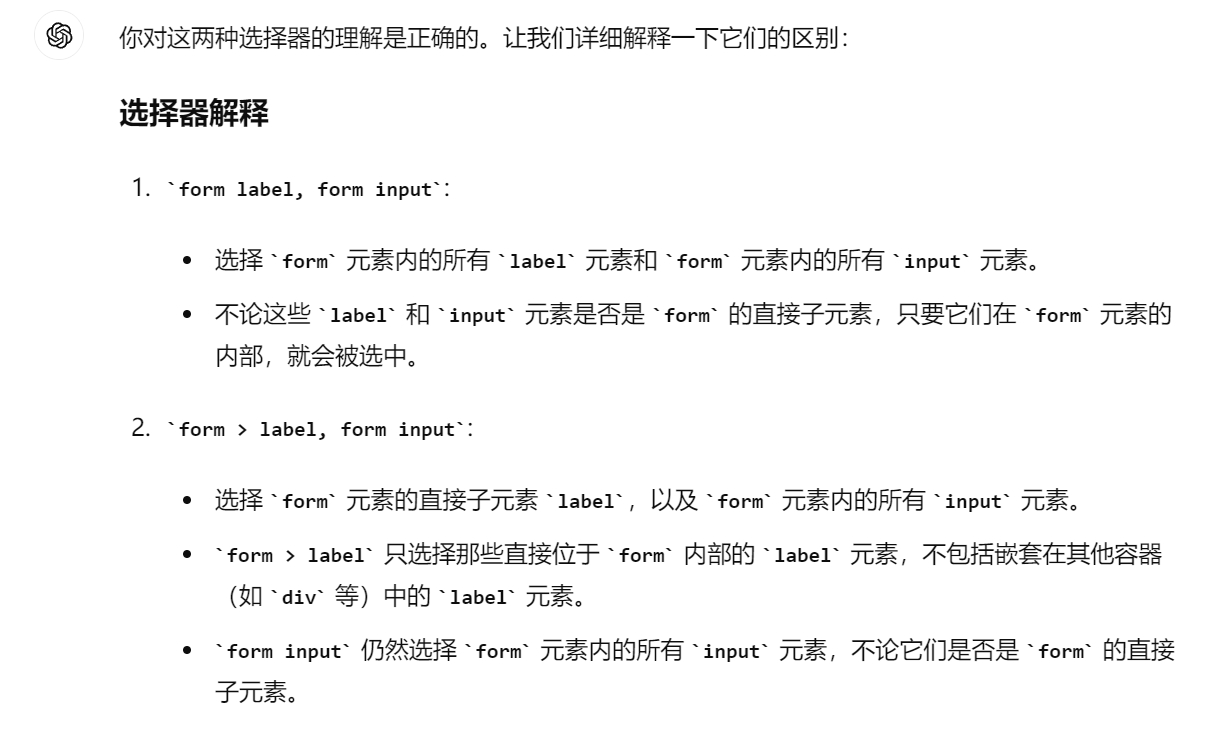
巨好看的渐变

一、根据元素位置应用CSS选择器

①使用包含选择符“ ”（就是一个空格）

e.g.选择li 标签内的em标签 应用CSS

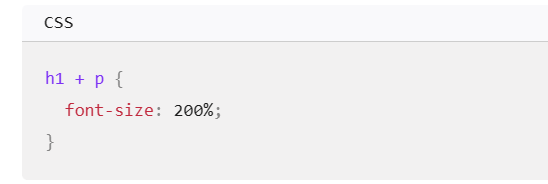
而且只要在里面就选中没有其他条件



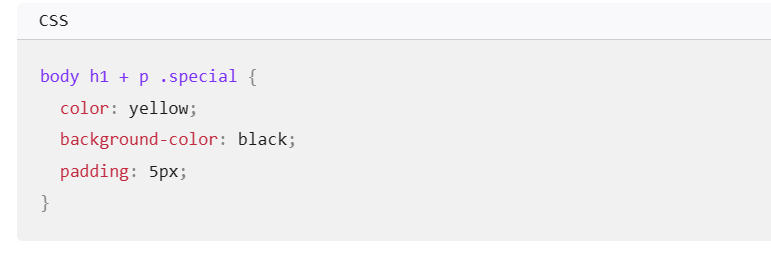
②使用相邻选择符“+”（一个加号）

在 HTML 文档中设置直接出现在标题后面并且与标题具有相同层级的段落样式

e.g.这样只会选择跟h1同级的，且直接出现在h1之后的p标签。



注意：必须是直接出现，但凡中间多了个h2这样的其他标签，那就废了



选择的是在 <body> 内，紧跟在 <h1> 标签后面的 <p> 标签内，具有类名 special 的元素。 但是你要是把.special前面的空格去掉就不一样了奥，选择的是：

在 <body> 标签内，紧跟在 <h1> 标签后面的 <p> 标签，并且这个 <p> 标签同时具有类名为 "special" 的元素。

**.box p：**选择所有类名是box的标签内的p标签

二、绝对长度单位和相对长度单位（[CSS 的值与单位 - 学习 Web 开发 | MDN (mozilla.org)](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Values_and_units#%E6%95%B0%E5%AD%97%EF%BC%8C%E9%95%BF%E5%BA%A6%E5%92%8C%E7%99%BE%E5%88%86%E6%AF%94)）

三、重要概念display：



就比如生成一个块级元素，但是display给他的属性是inline就会转化为内联元素，让他们在同一行内显示

选择器大考察：

body :not(section)#main > .content[data-type="article"]:nth-of-type(odd) .paragraph:first-child::first-letter + span > a[href$=".pdf"]:not([class~="external"])



联合选择器：选择”既有又有”

div#id { /\* 选择所有 div 元素，并且 id 为 id 的元素 \*/ }

.class1.class2 { /\* 选择同时具有 class1 和 class2 的元素 \*/ }

#id.class1 { /\* 选择同时具有 id="id" 和 class="class1" 的元素 \*/ }

#id.class1 { /\* 选择同时具有 id="id" 和 class="class1" 的元素 \*/ }

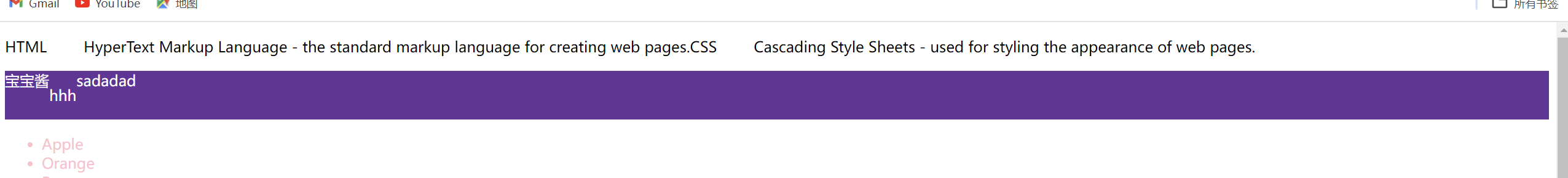
input[type="text"].class1 { /\* 选择所有 type 为 text 且 class 为 class1 的 input 元素 \*/ }，就是两个条件之间没有任何东西，直接连接起来

四、匿名块盒子（可以简单理解成没有标签的语句块）

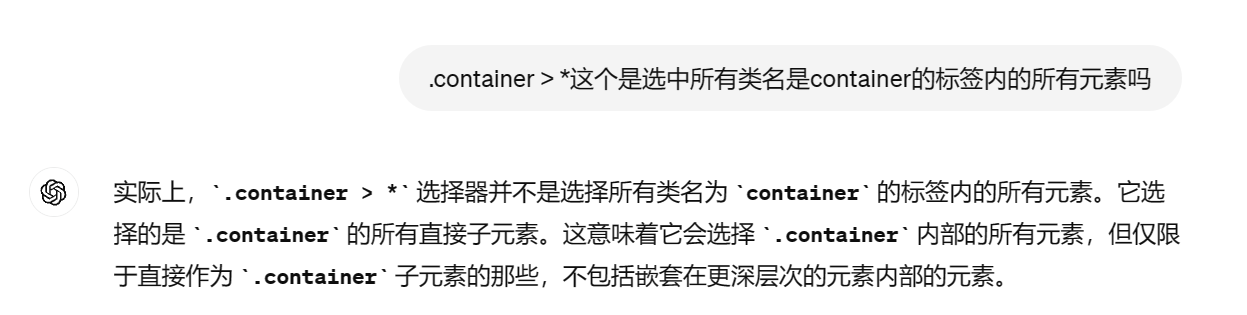
e.g.直接生成一个匿名盒子所以不能像<p>那样被CSS控制，又因为用的是弹性容器（display: flex）所以水平排列（flex是个块盒子，里面的元素横向排列（没设置的时候））



而且flex后面用的 > \*表示应用于flex类标签内的所有标签，但是去掉这个东西之后就会出现一个神奇的现象：一整个块级盒子全都被应用了（矩形）



原因就是，如果你不去设置是应用到div内包含的盒子（标签），CSS就会直接应用到这个div标签上，但是div又是一个块容器盒子，盒子可以理解成一个矩形，所以这个盒子所在的矩形区域就被应用了CSS，> \*表示选中这个选择器所指的标签的所有直接子元素（下面有图片）



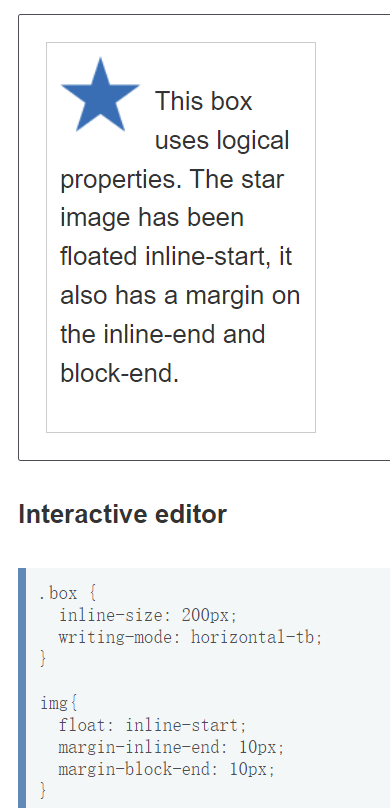
五、float

可以像display中的flex那样字改变标签的结构（左右还是上下）看下面的图片就是

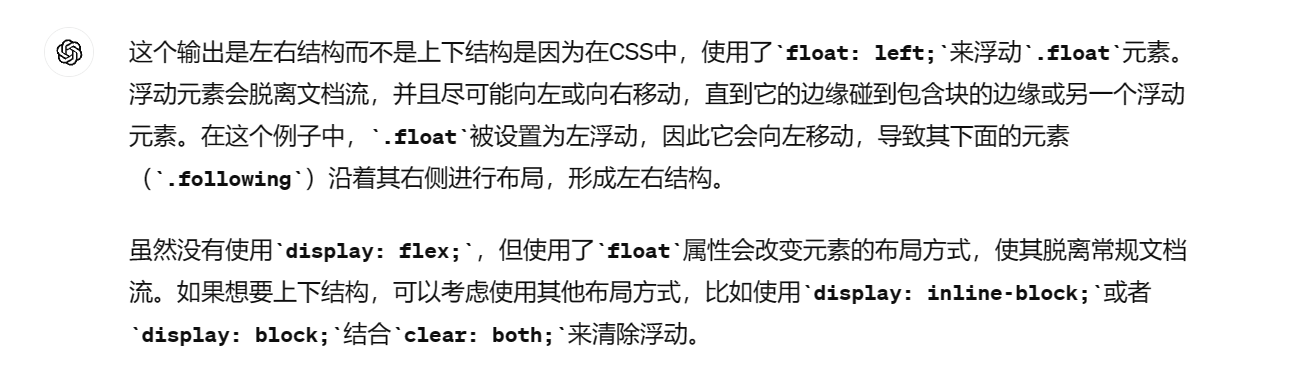
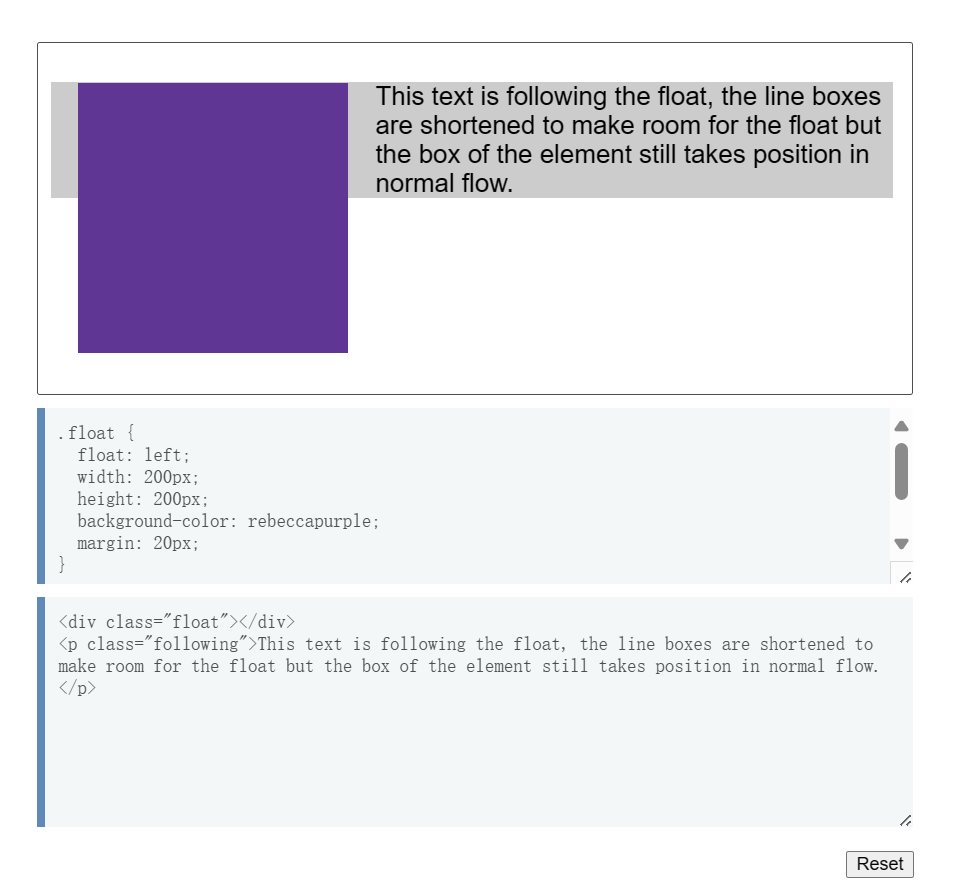
当把float: left;删掉就恢复了上下结构

Float就是让盒子浮动，从文档流中像水一样浮动到别的地方去，同时让别的元素环绕他

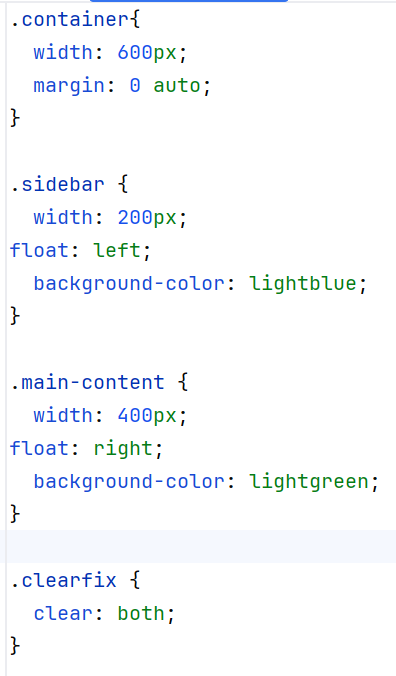
下图就是一个环绕的例子。



具体网页：[视觉格式化模型 - CSS：层叠样式表 | MDN (mozilla.org)](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Visual_formatting_model)

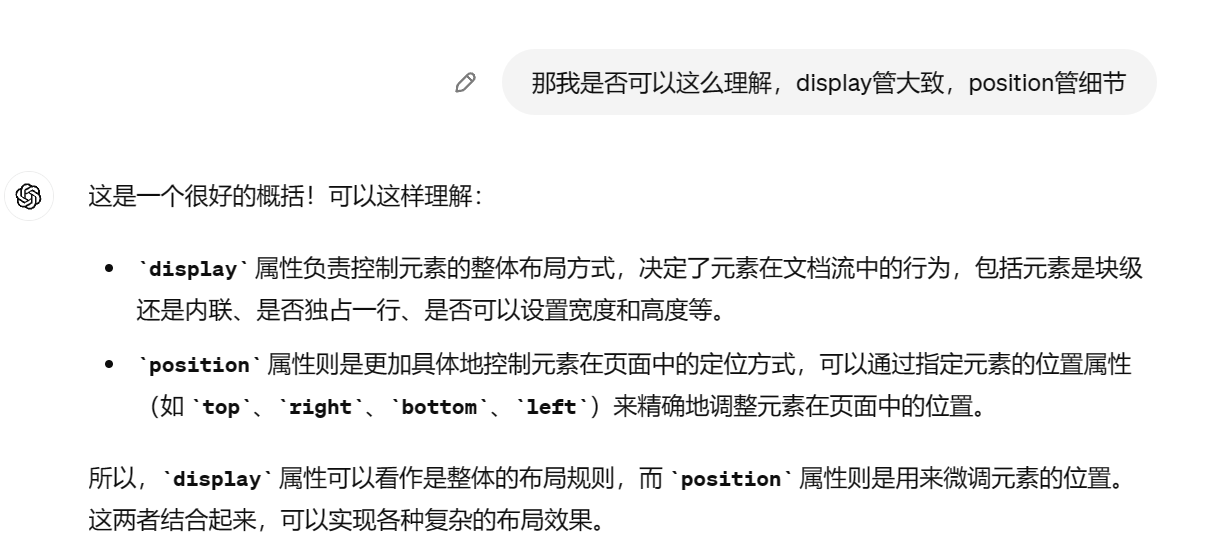


浮动完之后要用一下清除，保证后面的元素不被影响，如下图，给出了一个div来处理



六、position属性（注意区分这玩意儿的几个值到底有什么用，挺有意思的）

[position - CSS：层叠样式表 | MDN (mozilla.org)](https://developer.mozilla.org/zh-CN/docs/Web/CSS/position)

  
 The difference between display and position

七、渐变色彩<gradient>css里面的东西

[<gradient> - CSS：层叠样式表 | MDN (mozilla.org)](https://developer.mozilla.org/zh-CN/docs/Web/CSS/gradient)

八、运算符（选择器里面用）

e.g. article > p {

}这个会选择<article>的所有直接子标签<p>下图是例子以及说明



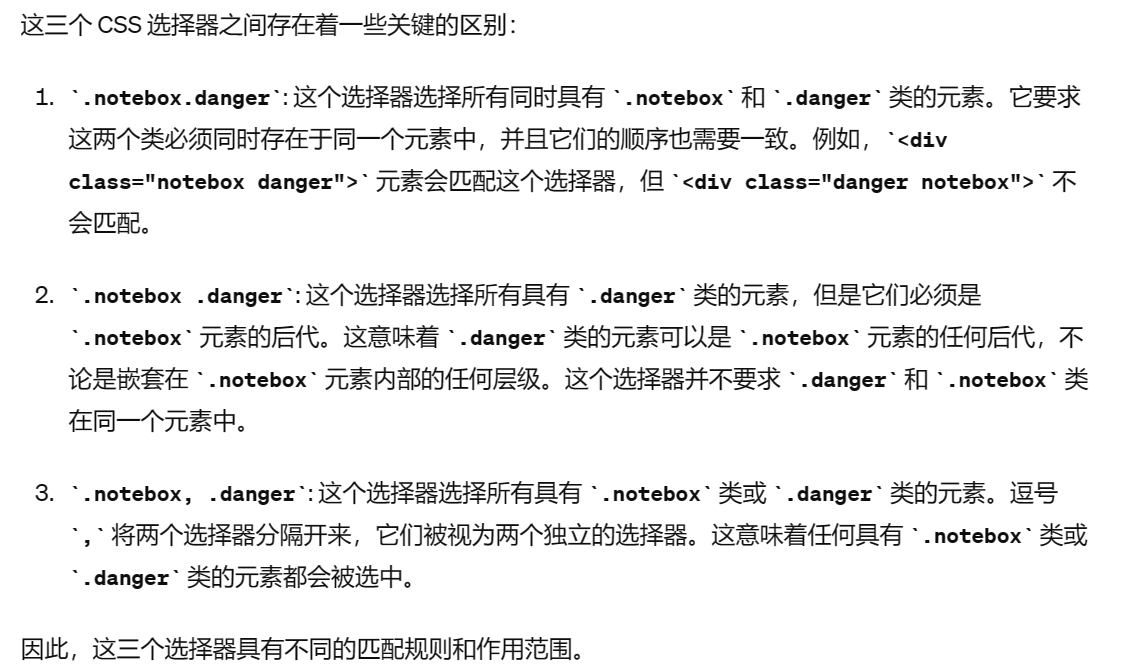
九、全局选择器的坑

①article :first-child {}这个选择的是<article>标签下的第一个子标签（无论什么类型）

②article:first-child {}这个选择的是，当且仅当<article>标签是他父元素的第一个子标签时才会被选中，例子在下面图片



注意区分下面三种



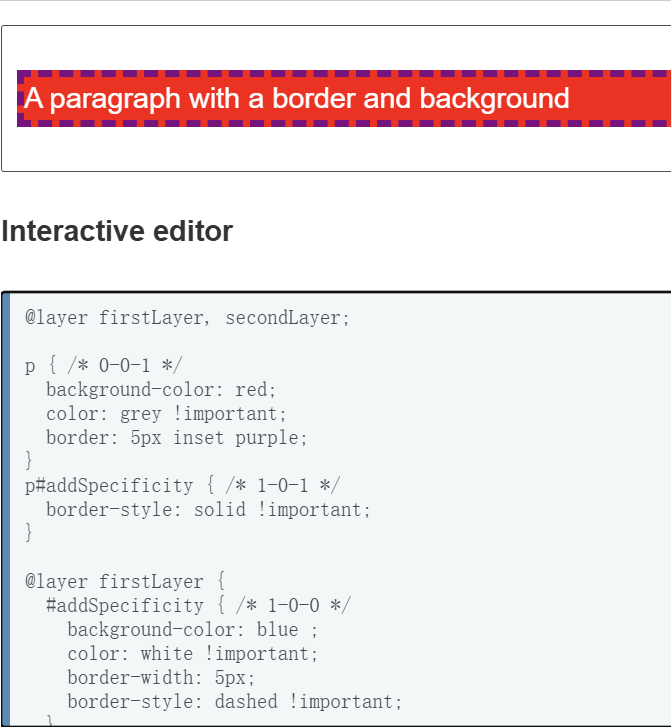
十、属性选择器（专门选择带有某些属性和值的标签）

[属性选择器 - 学习 Web 开发 | MDN (mozilla.org)](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Selectors/Attribute_selectors)

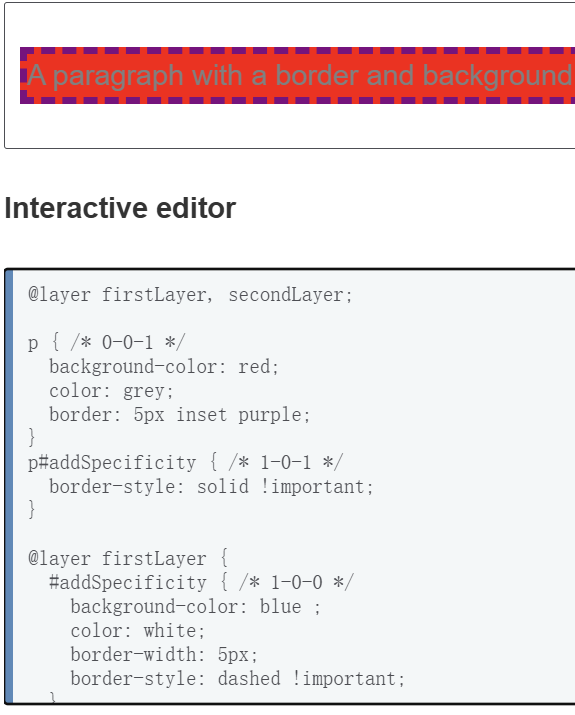
小测试：[技能测试：选择器 - 学习 Web 开发 | MDN (mozilla.org)](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Selectors/Selectors_Tasks)

十一、!important的优先级会反过来：

如果两个级联层以及有一个不分层的常规样式在同一种样式控制语句上面都用了!important那么本该常规样式生效，但是因为important所以导致级联层里面的生效



图中color 都有参数，并且都有important，但因为优先级倒置，所以文字应用了白色



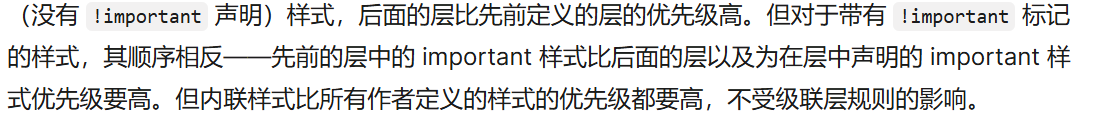
图中都没有important，因为常规样式比级联层优先级高所以文字应用了灰色



常规样式优先级 > 级联层，级联层内先定义的优先级高

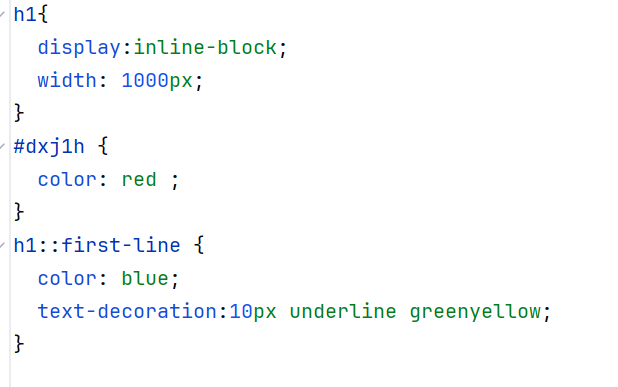


这个就是firstlayer > secondlayer 因为CSS会先处理前面的

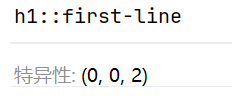


十二、

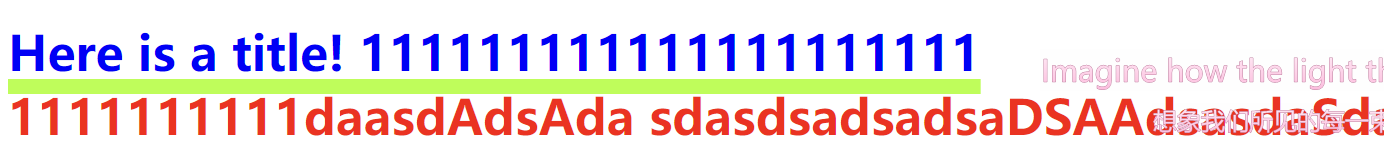
有些是有伪元素的属性会产生虚高的现象，众所周知伪元素在特异性计算里算1，但是有些时候会特别特别高比!important都高，以下是一个例子



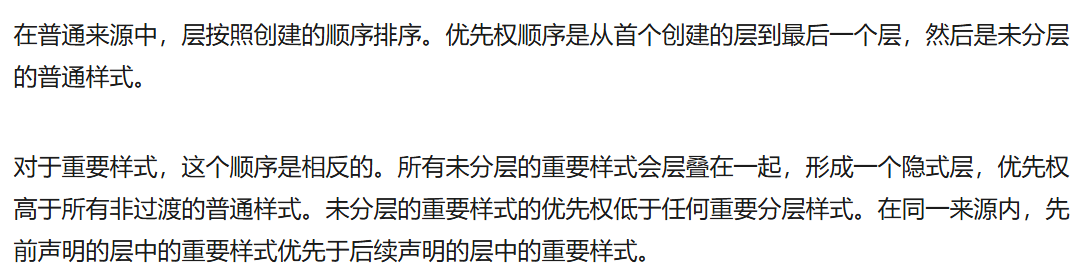
如图，给出了一个h1::first-line，但是这个特异性值是（0,0,2），所以肯定



比上面那个id选择器要低，按理来说这个里面的color会全部被red覆盖，但是不然，他的第一行依旧应用了蓝色（blue），哪怕加上!important也是这样，这就说明有些时候，伪元素的优先级特别高。

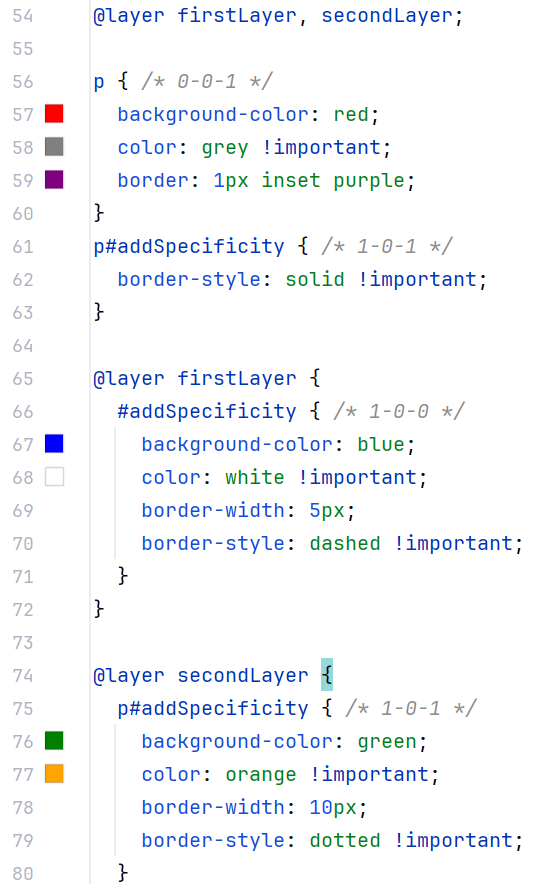


十三、层叠层优先级理解如下



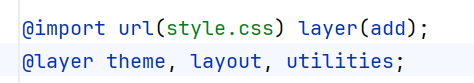
这里有一堆css，其中有一个层叠层，上面那一堆没归类的就是未分层的样式。上文的意思是，他们的优先级比分层的高（也就是比下面firstlayer、secondlayer高），所以该例子边框像素是10px,还有inset，purple。（未分层的会整合到一个未命名的层中然后放在最后）

后面一句话就是加了!important的（也就是未分层重要样式，分层重要样式）分层的优先权比没分层的高一点，所以该例子最后字体颜色是白色的。



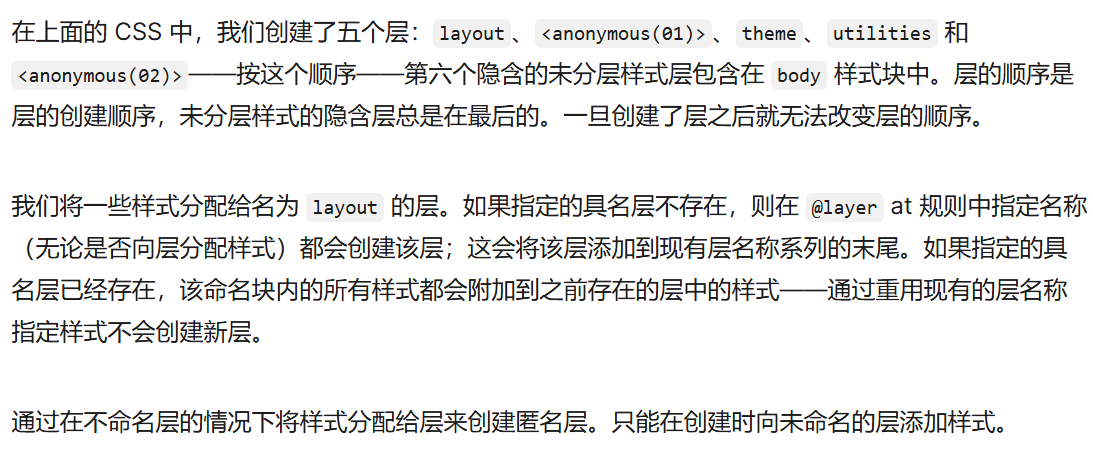
（未分层样式的隐含层总是在最后的，所以写在分层的前面优先权依旧高）

注意：哪怕有@import也遵守这个规则



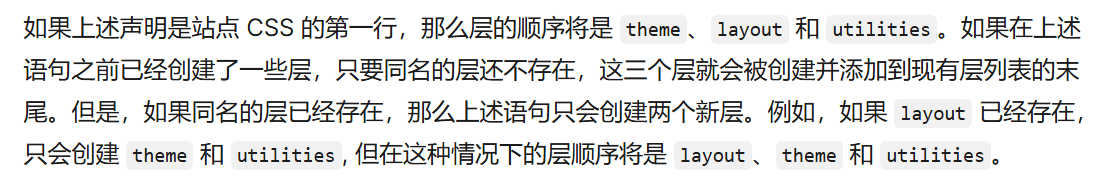
如图所示代码，这个生成的层从先到后是，add theme layout utilities

例子在这个网址：[层叠、优先级与继承 - 学习 Web 开发 | MDN (mozilla.org)](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance" \l "%E7%BA%A7%E8%81%94%E5%B1%82%E7%9A%84%E9%A1%BA%E5%BA%8F)

十四、层叠层重要性质  
 

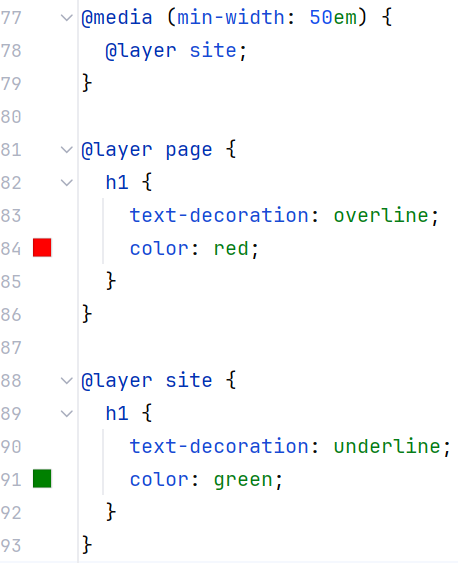
你声明的层叠层如果存在，那么会将你给出的CSS样式控制代码加入到已存在的层最后（

通过重用现有的层名称指定样式不会创建新层）



而优先权会根据该层是否已经存在而有所改变，如上图最后两行。

十五、用媒体查询更改声明层的顺序



如图，只有屏幕像素最小值比50em大，那么CSS会先声明一个site层，也就是把site从原来的第二，通过一次声明变成第一，所以生效的是page层。

十六、嵌套层叠层重要理解：



这个就是，components层中如果第一句话的components-lib.css，已经有了一个narrow层，那么第二句话的narrowtheme.css里面的css语句就会直接加在这个层的末尾。“.”符号用来创建一个嵌套层。

十七、替代盒模型（很方便的就可以计算出来高度和宽度，较常用）

你width，height设置多少，这个盒子宽高就是多少，

一般比较常用，下面给出预处理代码

\*{

box-sizing: border-box;

}

CSS

\*,

\*::before,

\*::after {

box-sizing: inherit;

}

十八、外边距设置规则（margin）  
 如果多个侧边距语句应用在同一处（作用效果重叠，比如同时设置margin-bottom）

1. 两个同号，那么取绝对值最大的那个
2. 异号，取正值，然后加上那个负值，e.g.一个margin-bottom50px另一个-margin-top-70px那么会先最底下间隔50px然后再加上-70px，最后形成的就是，下面那个盒子会从底下跑进上面那个盒子20px

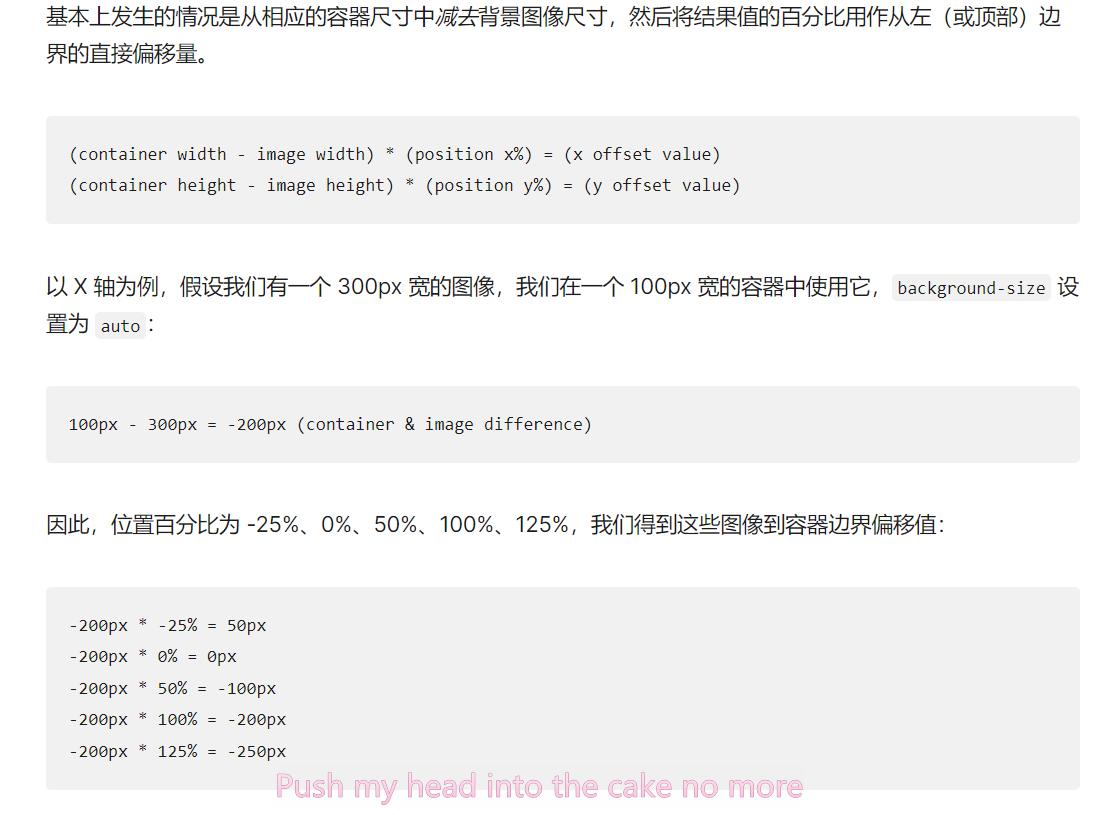


十九、CSS的简写属性（很好用，哪怕用不来起码的看得懂）

[CSS 的简写属性 - CSS：层叠样式表 | MDN (mozilla.org)](https://developer.mozilla.org/zh-CN/docs/Web/CSS/Shorthand_properties)

二十、百分比的background-position位置计算公式

<https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-position>



先按容器大小减去图片大小（以X轴为示例），图像300px，容器100px，所以得到

-200px，然后如果background-position中有个-25%那就用这个-200px去乘-25%得到50px,这个就是偏移量，表示图片左边界距离容器左边界50px（因为容器100px所以图片左边界在容器中间）

CSS渐变生成器：<https://cssgradient.io/>

Background: color size(contain or cover or 百分比值) position image(引入图片，渐变) repeat attachment（页面或元素滚动时背景是否跟着一起动）

# 二十一：border-radius

# /：用于分开水平与垂直方向

# 

# border-radius: 10px; /\* 所有四个角都是 10 像素的圆角 \*/

# border-radius: 10px 20px; /\* top-left 和 bottom-right 是 10px，top-right 和 bottom-left 是 20px \*/

# border-radius: 10px 20px 30px; /\* top-left 是 10px，top-right 和 bottom-left 是 20px，bottom-right 是 30px \*/

# border-radius: 10px 20px 30px 40px; /\* 分别为 top-left、top-right、bottom-right 和 bottom-left 角指定不同的值 \*/

简而言之，他跟很多元素的简写方式都一样都在那个CSS简写属性里面有写

二十二、background-attachment  
scroll 盒子内部的文本动，背景不动，整个盒子动，背景动

Fixed 正如其本意，所以背景死活不动

Loca 正如其本意，本地，所以跟着文本一起动，文本不管怎么动，他都跟

<https://developer.mozilla.org/zh-CN/docs/Web/CSS/background-attachment>

# 二十三、处理不同方向的文本

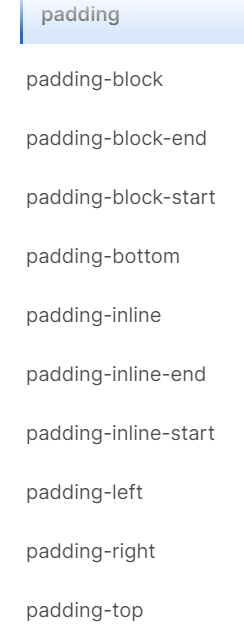
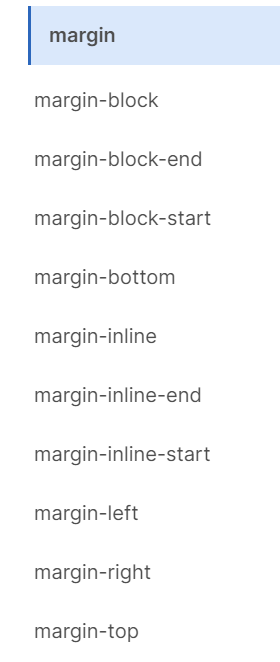
# e.g.writing-mode: vertical-lr; 垂直排，文字从左到右（文字都是block方向上）

要点：width、height、margin、border、padding都会被映射成新的

其中要理解文本被转向之后，我们就应该以block和inline来看待height和width。

Width变成了inline-size height变成了block-size margin要看具体在哪里，然而只分start和end，根据文本开始在左还是右来应用。纵向默认从上往下，上是start

margin padding border都是具有相同的拓展名规范的

# 二十四、object-fit

# object-fit跟background-size很像只是一个用于HTML元素里面的img，另一个用于background-size里面的用于调整背景位置的，他们两个支持的值也特别像，就比如都有cover。

# 注意区分：

# object-fit: fill;` 会拉伸图像以填充元素的内容框，不保持原始图像的纵横比，导致图像可能会被拉伸或压缩以适应元素的大小。

# object-fit: cover;` 会保持图像的纵横比，并尽可能填充整个元素，但可能会裁剪图像来适应元素的尺寸，以确保元素完全被覆盖。（也就是说保持等比放大然后截取，background-size：cover也一样）

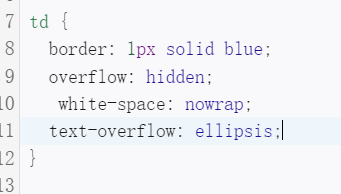
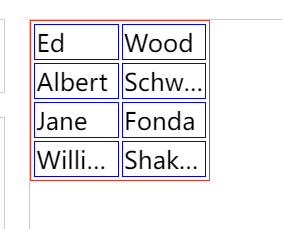
# 而且用的时候也不能单单就一个object-fit可能还要用到截图里的另外两个（要搭配height和width一起用）

# 用在img这个选择器里面

# <https://developer.mozilla.org/zh-CN/play>这个网站是一个实例，也是在img里面用了width和height

二十四、text-overflow: ellipsis;，让表格中溢出的元素显示成省略号

一定要跟以下这几个一起使用才会生效。（width有些时候也可不搭配，但是另外两个一定要）

 你看，没width也可以实现。



#### 二十五、:nth-child这种伪类下面举例说明

就比如li:nth-child（-n+3）这种东西表示的是，从父级元素里面找子元素li,而且找的是第3，第2，第1（因为n取0,1,2时-n+3 != 0所以不取n=3）

二十六、white-space

两大作用：

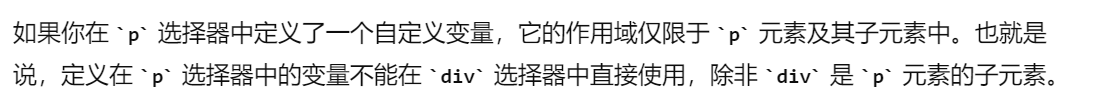
* 空白字符是否[合并](https://developer.mozilla.org/zh-CN/docs/Web/CSS/white-space#%E5%90%88%E5%B9%B6%E7%A9%BA%E7%99%BD%E5%AD%97%E7%AC%A6)，以及如何合并。
* 是否换行，以及如何换行。

二十七、text-shadow，给文字加阴影，正如上面写的一样

值顺序：一定是offset-x ,offset-y，blur-radius，其中blur-radius可选，color可以放在这几个的最前面或者最后面，不可放在中间！！！（其中offset-x ,offset-y必选，要是你给他弄个0然后又没有其他参数，还不如不写）

二十八、css自定义变量

网站里面都有，但是在此特别声明的是，:root(也就是html)这个选择器里面的自定义变量才有全局属性，但凡是一个特定选择器内的变量，都只能在同一个特定选择器内使用。



<https://developer.mozilla.org/zh-CN/docs/Web/CSS/Using_CSS_custom_properties>

二十九、可以应用模糊的属性：  
text-shadow、box-shadow、filter中的drop-shadow

filter(过滤器)很重要，可以应用很多属性

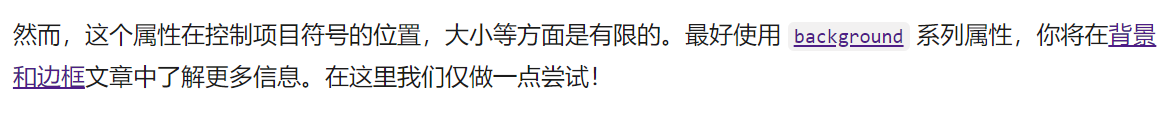
三十、列表可以自定义样式（部分）

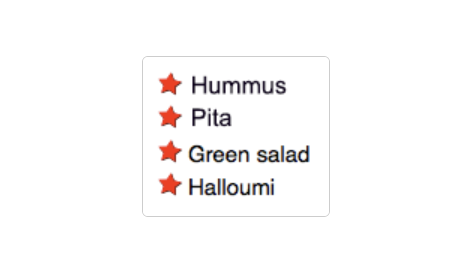
list-style-type、list-style-image、list-style-position(inside、outside)

他们可以简写在list-style里面



其中list-style-image最好用background系列属性，他这个背景改变的是无序列表前面圆圈的样式



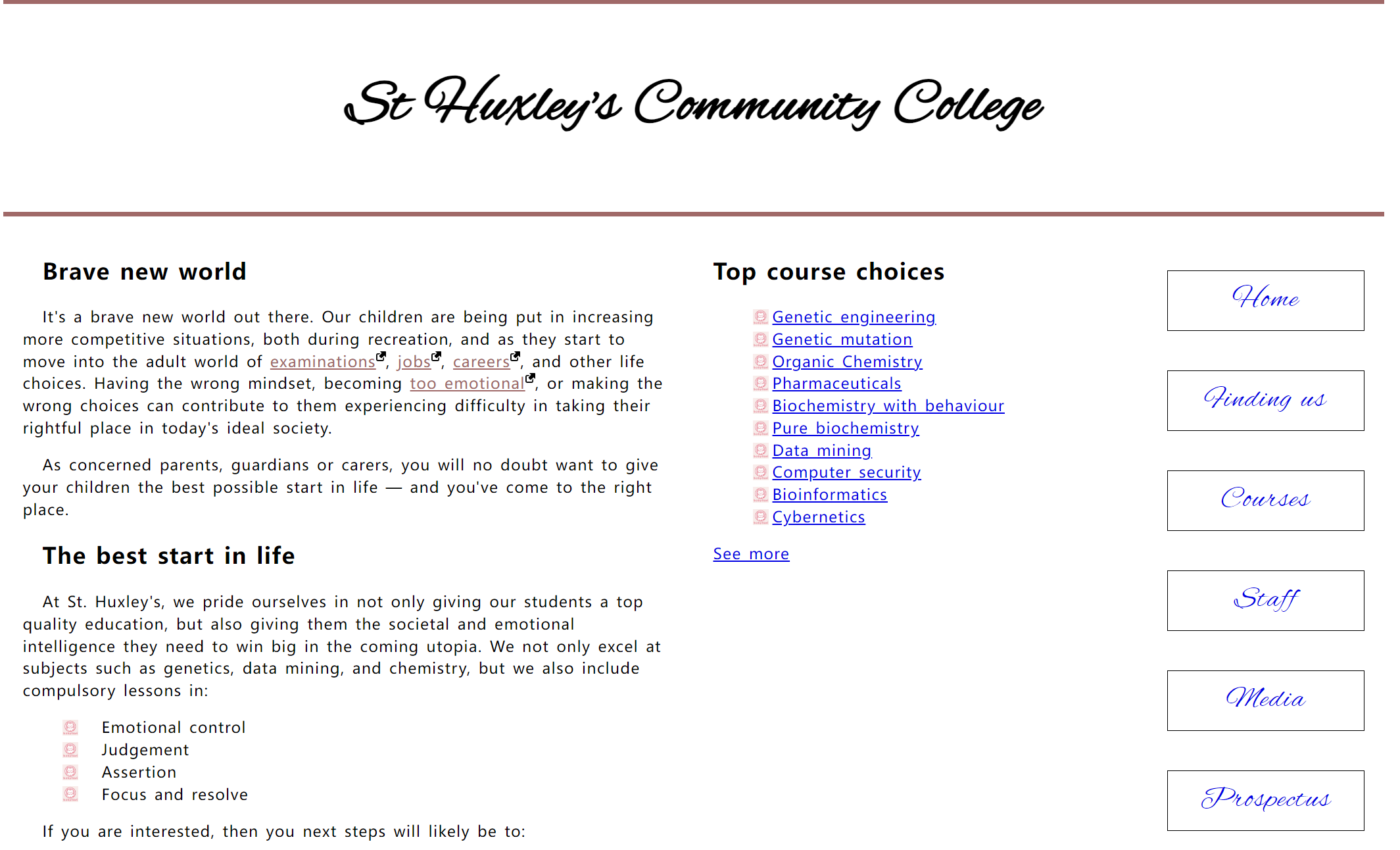
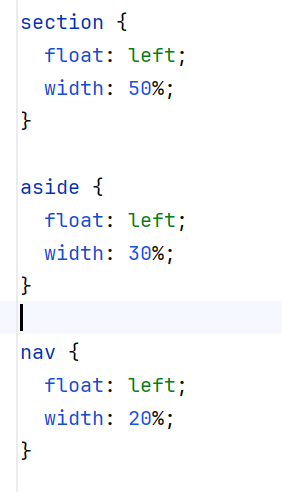
 

三十一、样式化超链接

默认设置基本都是text-decoration: none; outline-color: transparent 有这两句话，是防止默认的属性影响我们自定义的属性，当然如果需要默认属性也可以不加。

三十二、CSS用float排版说明：

问题：nav用的是float：left，但是他却出现在网站的最右边（web字体等作业）



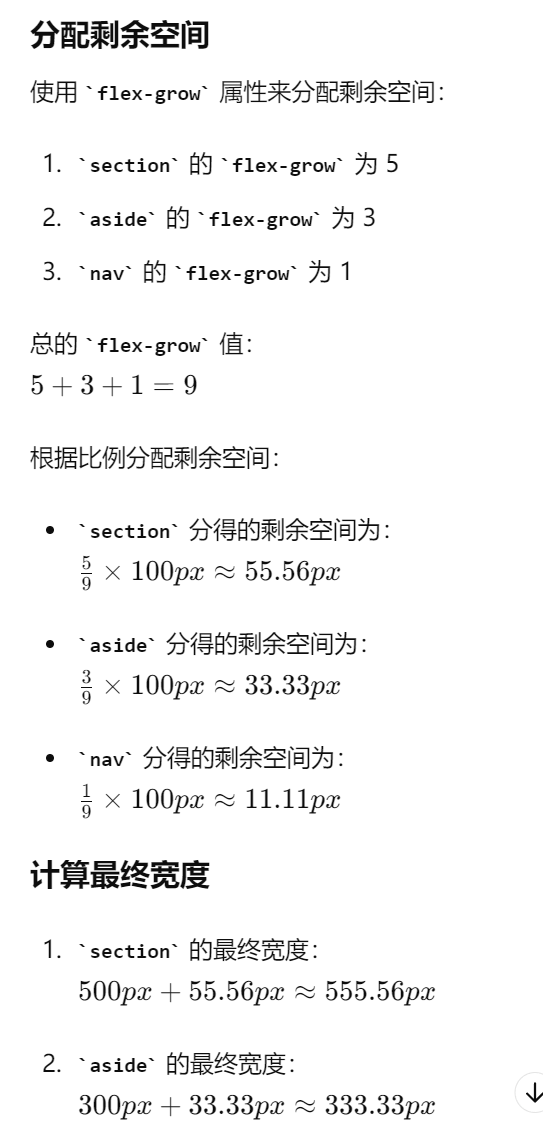
解决：因为float是浮动，前面的section 和 aside 都被浮动到了左边，只有右边的位置留给nav然后float：left就自然而然的只能在最右边，float：right的效果在这里也是一样的。

三十三、flex-grow，flex-basis，flex-shrink，flex-wrap，justify-content，align-items详解  
 flex-grow：掌管剩余空间如何分配

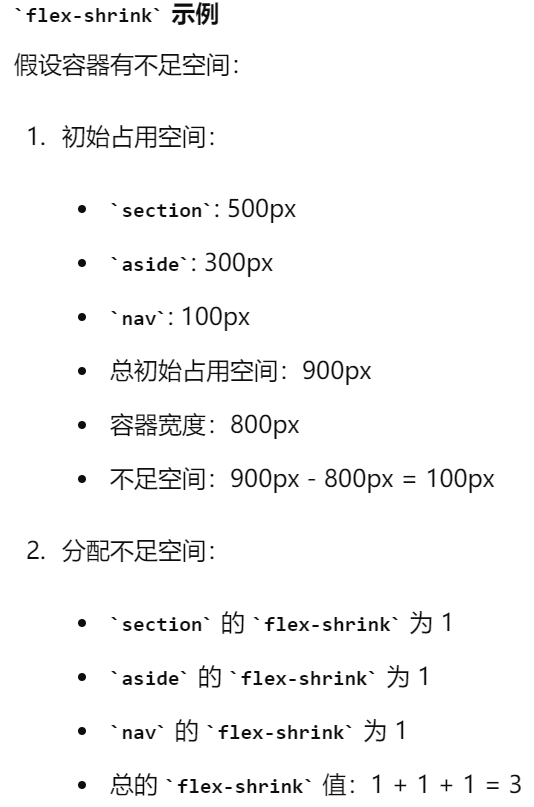
flex-shrink：掌管若超出盒子总大小，如何收缩各个盒子

flex-basis：掌管初始空间如何分配（类似于width）

先按初始空间分配，如果有多余，再按flex-grow来按各自的值进行分配并加到每一个盒子上，下面是例子。



盒子空间不足时，就需要用到flex-shrink，以下是应用实例（代码跟上面一样）  
  

但是flex-shrink: 0是个霸王属性，表示该元素无论何时都不参与收缩

Flex-wrap：这个来决定这个盒模型里面的元素是否全部都要包裹进盒子里面，说白了就是允许在快要溢出容器的时候换行处理，默认是no-wrap（也就是不允许换行）

Flex-flow：简写[flex-direction](https://developer.mozilla.org/zh-CN/docs/Web/CSS/flex-direction) 和 [flex-wrap](https://developer.mozilla.org/zh-CN/docs/Web/CSS/flex-wrap)，默认是flex-flow：row，wrap（与刚刚讲的有那么点不同）

Justify-content：控制主轴方向上的flex各个元素的位置（在一定程度上不需要用上面那三大件）

Align-items： 控制交叉轴方向上的各个元素的位置（其中可以使用align-self对某一个元素进行调整）

Flex可以进行嵌套，详细例子看<https://developer.mozilla.org/zh-CN/docs/Learn/CSS/CSS_layout/Flexbox#flex%EF%BC%9A%E7%BC%A9%E5%86%99%E4%B8%8E%E5%85%A8%E5%86%99>即可

:nth-of-type ：可以选择同类元素的某一个，某几个

三十四、grid（网格布局）



单一数值是从当前这个列或行到下一个列或行

三十五、flex，grid总共涉及的六个容易搞混的属性

align-content justify-content

align-items justify-items

align-self justify-self

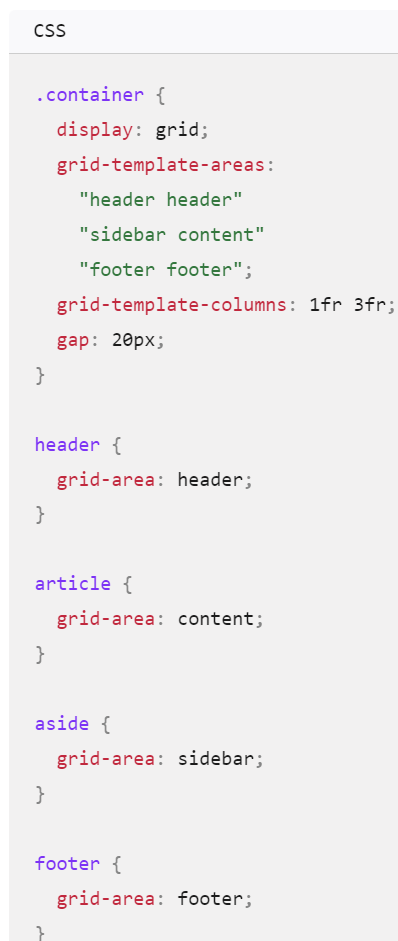
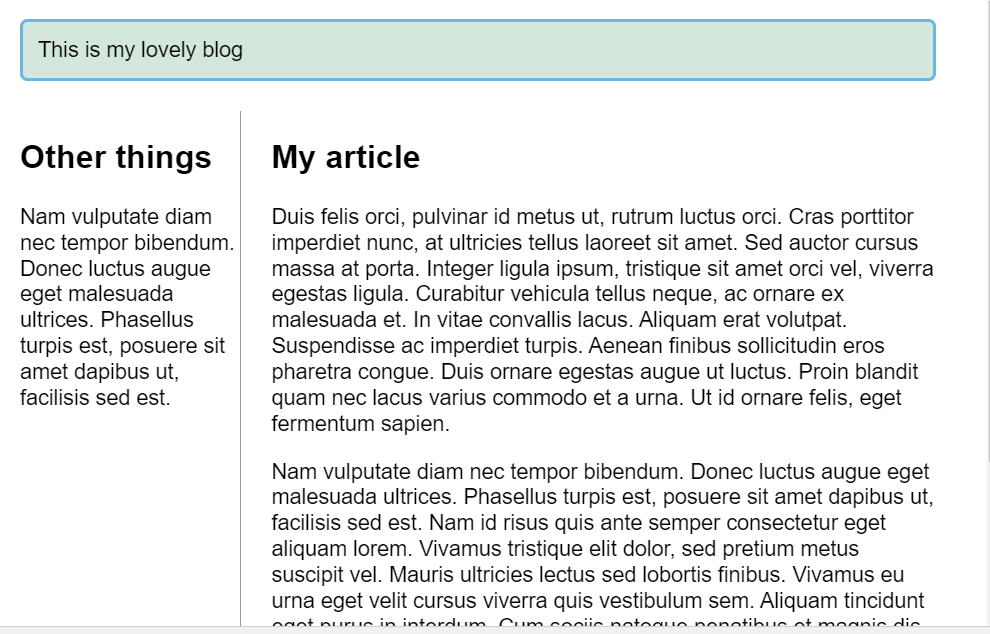


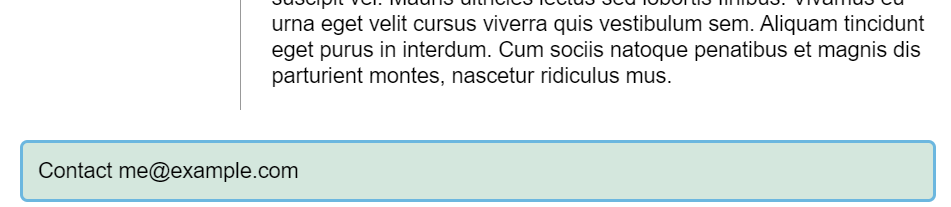
还有place，是对align和justify的合成

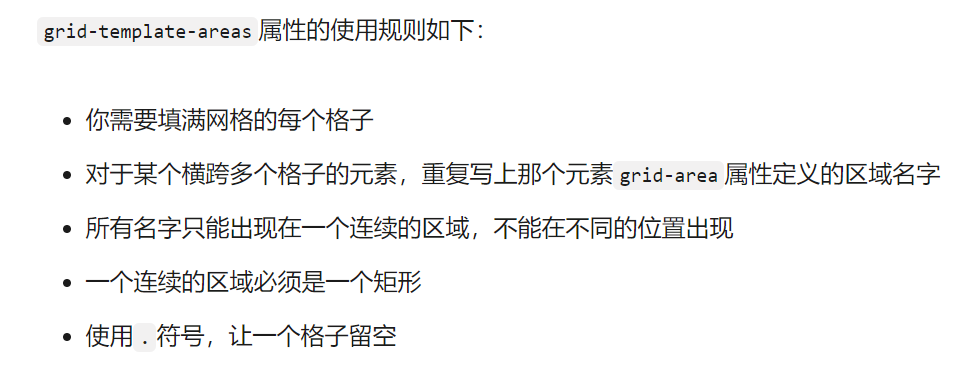


三十六、grid-template-area

这个属性特别神奇，你只需要给各个部分取上名字，然后放好，最后在每个html标签的选择器那里使用对应的名字，这些标签就会被放到对应的地方上去。

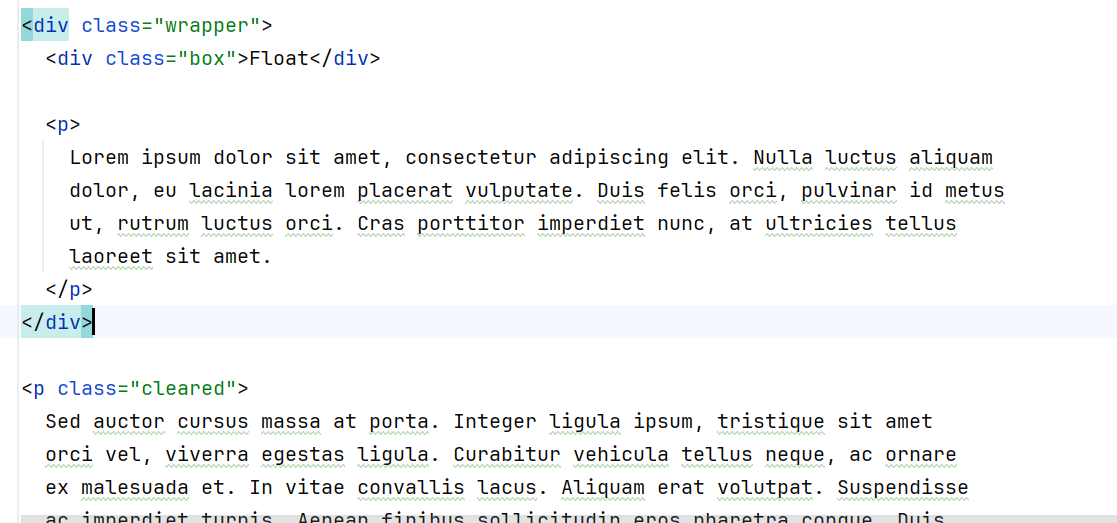
最后会变成如图这个样子，说白了就是会按名字的位置对应



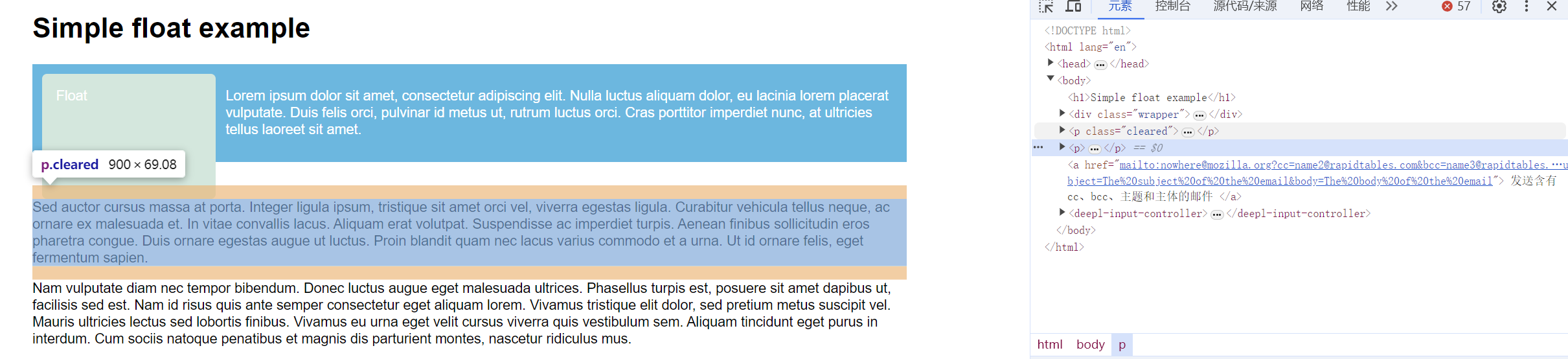
使用规则：

三十七、关于float布局的一些问题以及解决，以及BFC介绍，解释

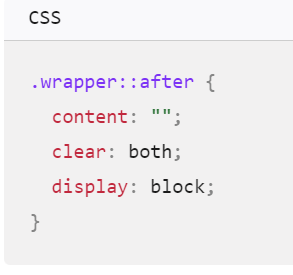
有些时候如果一组元素（存在父级子级关系的）的某一个应用了float，那么可能导致这一组元素之外的元素在面对clear（清除浮动的时候出现清除不干净的问题）例子如下



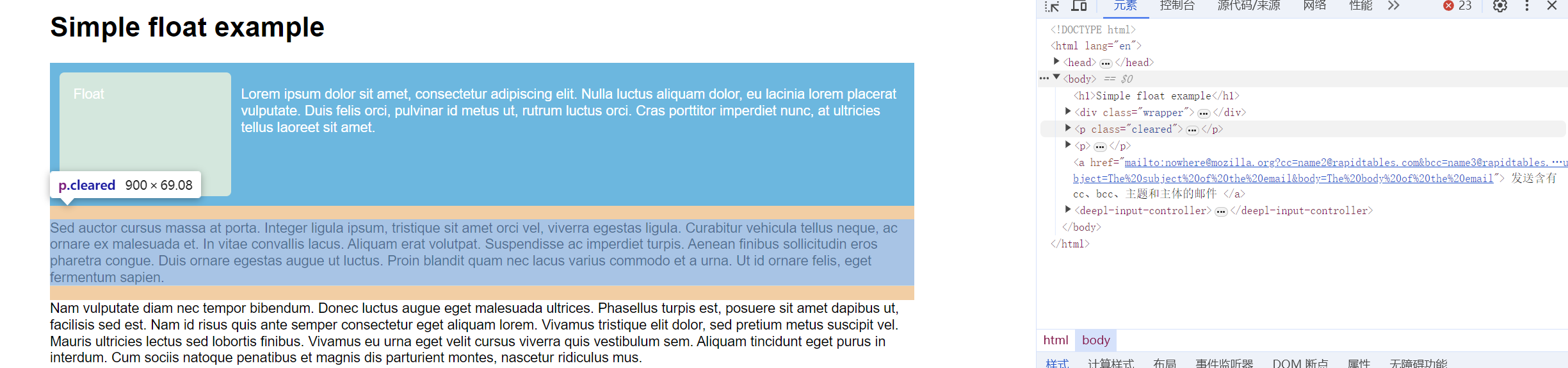
这里有一个盒子，包裹这两个元素，那么这个.wrapper就是个父级元素，暂且称之为元素组，这时候对下面的元素p进行clear会发现有如下问题：



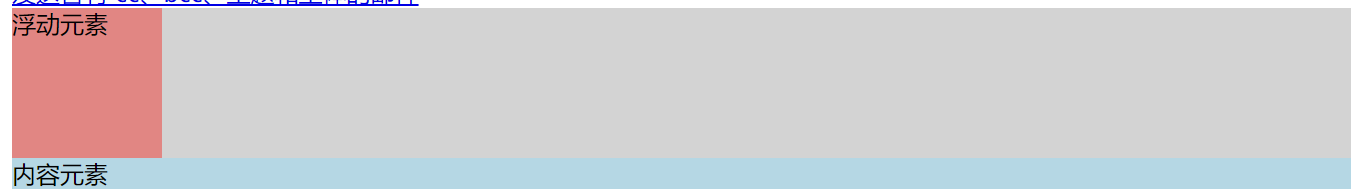
很明显，clear（清除浮动没有清除干净）。这时就要用到一个小技巧叫做clearfix



也就是在这个父级元素之后添加一个没有任何东西的块盒子，注意一定是块盒子！！！，然后就会成功彻底清除（下图所示）



BFC:区块格式化上下文，我们先来看一个例子

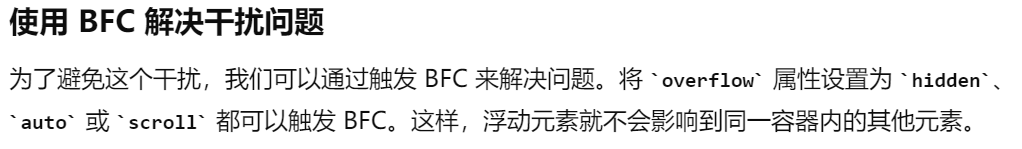


如图两个块盒子只是设置了背景颜色，各自的大小，并没有进行布局

但是当我们给，浮动元素那个块盒子设置成float：left的时候，就出现问题了



如图，由于设置了浮动，内容元素这个盒子升了上去，但是却干扰了他俩所在的那个盒子本来的样子，也就是背景颜色消失了（跟上面呢个例子一样，背景元素消失），因为浮动元素影响到了容器内的其他元素，这时就要用BFC消除干扰

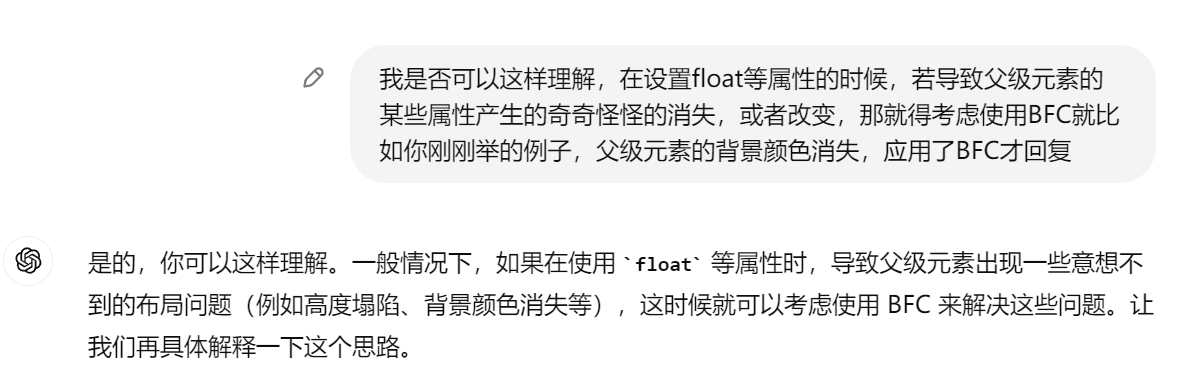


（使用BFC有很多方法，挑合适的或者顺手的即可），然后我们可以发现如下图的情况



原先盒子的背景颜色回来了，干扰成功消除

BFC的工作岗位：

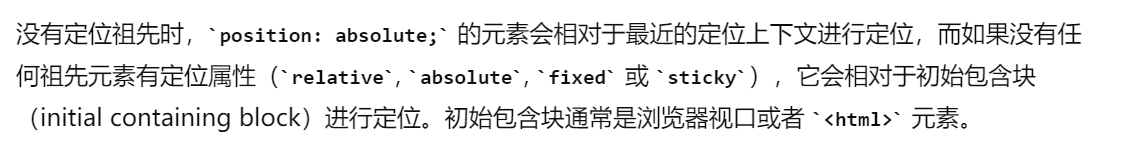


当然，创造BFC的方法有很多，但是较为现代且无副作用的方式是display: flow-root;

（在父级元素中声明，也就是用了float的那个元素的父级元素）

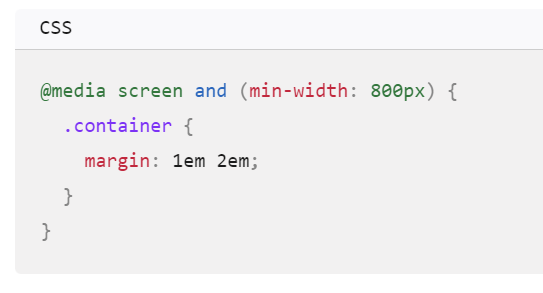
三十八、改变absolute所相对的盒子

将父级元素的position设置成relative。本来absolute是相对于<html>元素也就是整个文档流的位置，但是有先祖定位也就是父级元素的position被设置成除默认以外的东西，那么这个absolute相对的东西就会改变

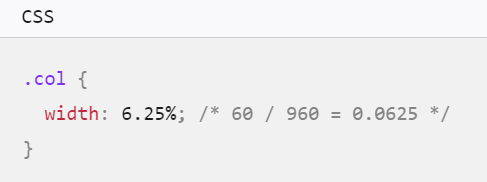


三十九、响应式设计大总结

1. 媒体查询



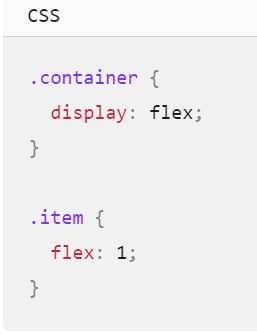
1. 灵活网格（采用百分号单位的宽度设置）



1. 多栏布局：使用column-width来固定每一列盒子的宽度，浏览器会更具视口来增添或者减少列数。



1. 伸缩盒（就是使用弹性盒子flex）



1. 响应式图像，可以看HTML对于这个是如何操作的，picture，img的srcset和sizes属性
2. 响应式排版，通过媒体查询，对不同的屏幕采用不同的CSS语句，主要原理是层叠层



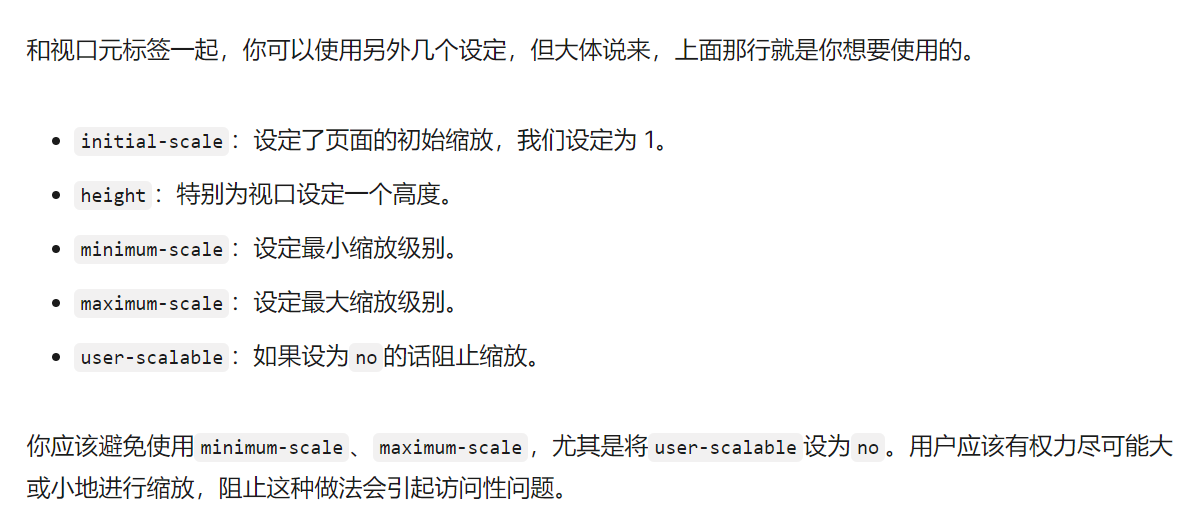
7．永远都不要只用viewport来设定文本，可以搭配calc函数来动态设置



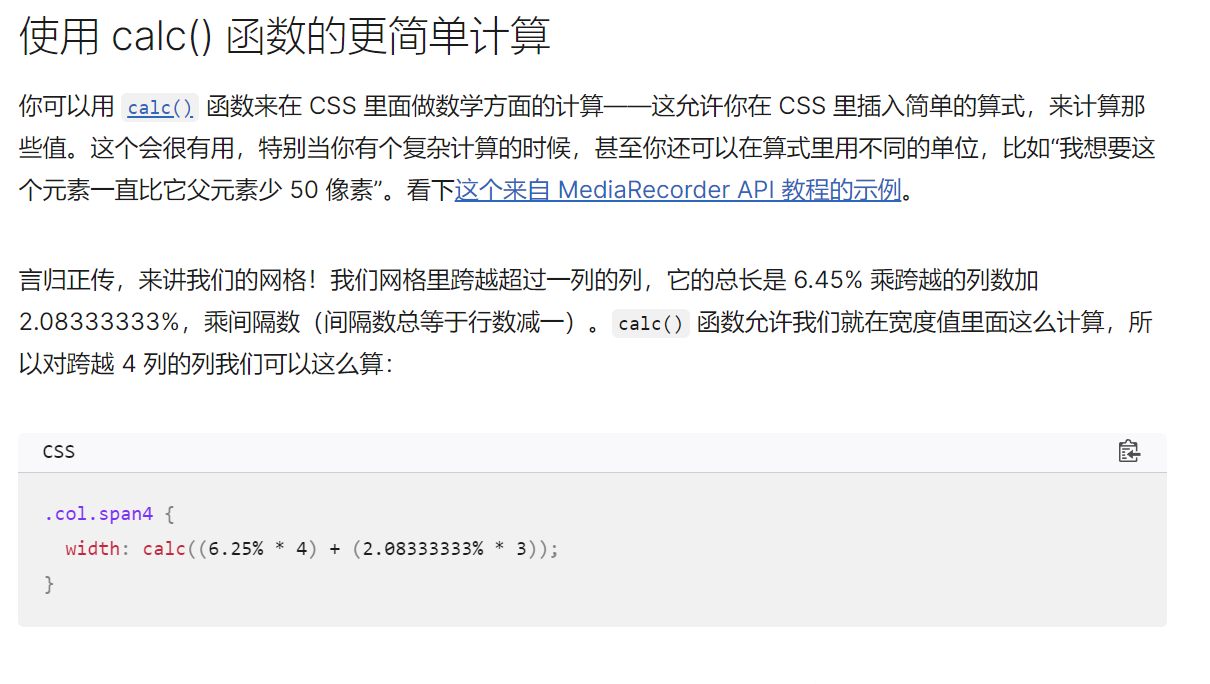
8.视口元标签（这句话基本必须写！！！！）

<meta name="viewport" content="width=device-width,initial-scale=1" />

这个元标签告诉移动端浏览器，它们应该将视口宽度设定为设备的宽度，将文档放大到其预期大小的 100%，在移动端以你所希望的为移动优化的大小展示文档。

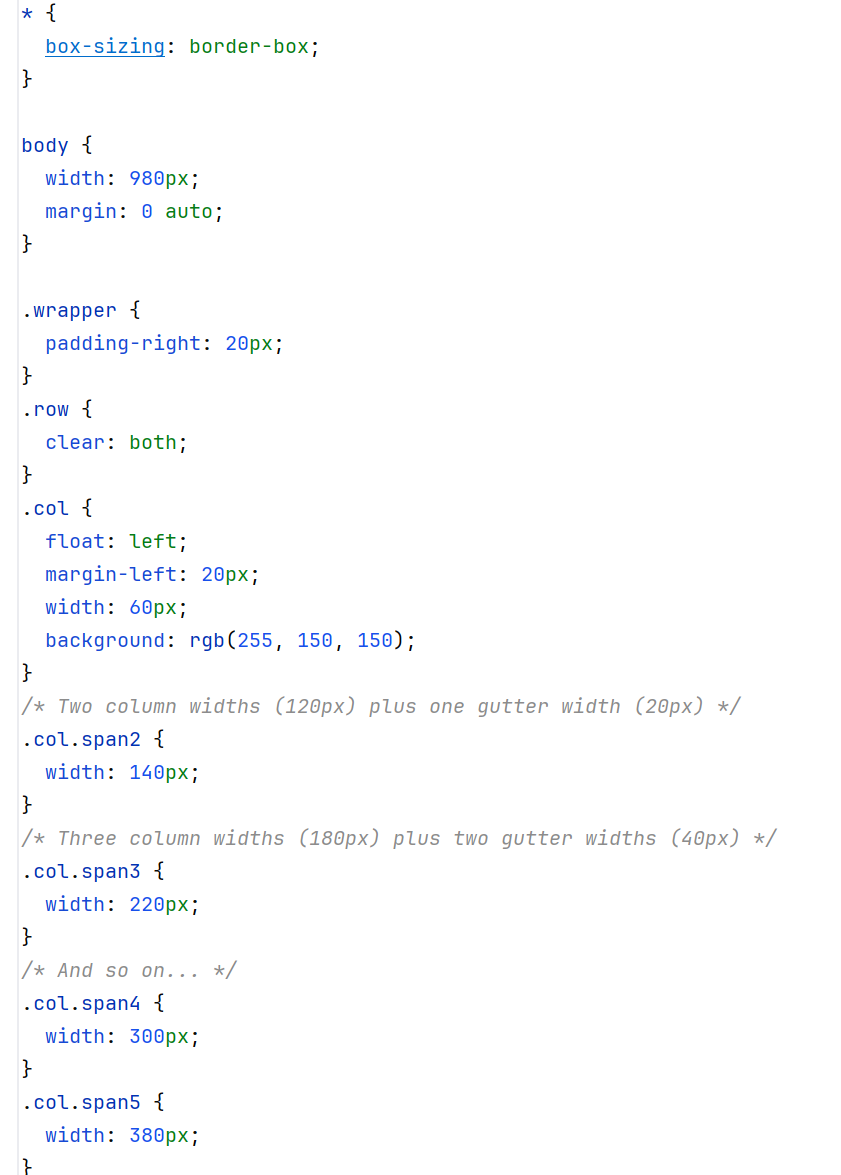


9.可以用calc函数进行计算大小



四十、传统网格创建

主要是用float以及设置每一个元素的宽度来形成网格



如图，设置不同的元素不同的宽度，以及使用float控制元素形成网格布局

可以去web学习的文件夹里面查看实例

D:\Web学习\HTML学习\传统网格创建实例（float）

四十一、特别注意！！！！

Float不可以跟position: sticky一起使用，会导致sticky失效，如果失效，可以试着采用grid等其他布局方法

四十二、flex默认属性



四十三、gap属性可以用于flex、grid以及多列布局