## 盒模型？怎么设置标准盒模型、IE盒模型? box-sizing属性？（作业帮）

有两种盒子模型：W3C标准盒模型、IE盒模型

盒模型都是由四个部分组成的，分别是content（内容）、padding（填充）、border （边框）和margin(边界)。

标准盒模型和IE盒模型的区别在于设置元素的width和height时，所对应的范围不同。标准盒模型的width和height属性的范围只包含了content，而IE盒模型的width和height属性的范围包含了content、padding和border, 指的是content+padding+border。

我们可以使用CSS3新增的的box-sizing属性来设置元素的盒模型类型，默认值为content-box，即标准盒模型，如果将box-sizing设为border-box则用的是IE盒模型。

（如果在ie6，7，8中DOCTYPE缺失会将盒子模型解释为IE盒子模型。若在页面中声明了DOCTYPE类型，所有的浏览器都会把盒模型解释为W3C盒模型。）

## 块级元素和行内元素（区别）（携程一面）

块级元素的特征：

* 每个块级元素都是独自占一行。
* 元素的高度、宽度、行高及顶部和底部边距都是可以设置的。
* 元素的宽度如果不设置的话，默认和父元素的宽度一致（父元素宽度100%）

常见的块级元素有：**<div>、<p>、<h1>...<h6>、<ul>**、**<li>**、<dl>、**<table>**、<address>、<blockquote> 、**<form>**

行内元素（inline）的特征：

* 每一个行内元素和别的行内元素排列在同一行里，直到一行排不下了，才会换行。
* 行内元素的高度、宽度、行高及顶部和底部边距不可设置。
* 行内元素的宽度就是它包含的文字或图片的宽度，不可改变。

常见的行内元素有：**<a>、<span>**、**<i>、<em>、<strong>、<label>**、<br>、<q>、<var>、<cite>、<code>

补充：内联块元素/行内块(inline-block)元素：

* 和其他元素都在一行上；
* 元素的高度、宽度、行高以及顶和底边距都可设置。

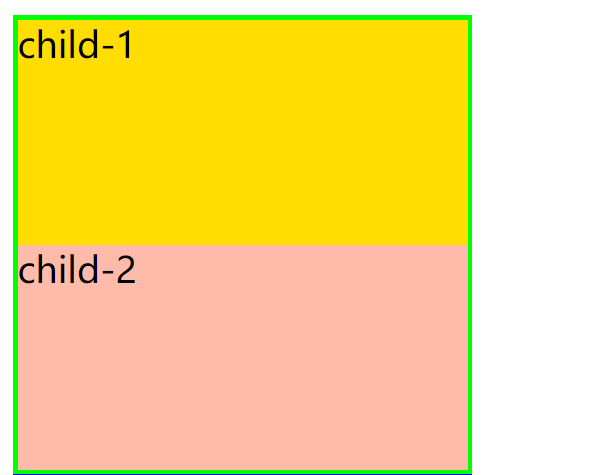
如：<img>、<input>标签

## display属性值有哪些？display: none和 visible:hidden的区别？（美团一面）

## position属性的值？relative 和 absolute 定位原点是？

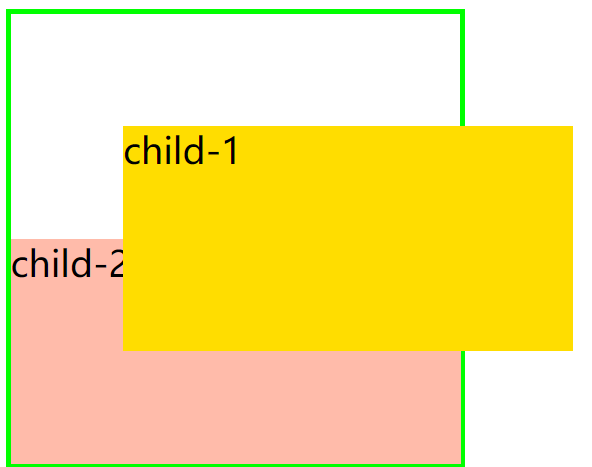
position属性值常见的有5个：

1. **static**（静态）：默认值。没有定位，元素出现在正常的流中当前的布局位置（忽略 top, bottom, left, right 或者 z-index 声明）。不脱离文档流。

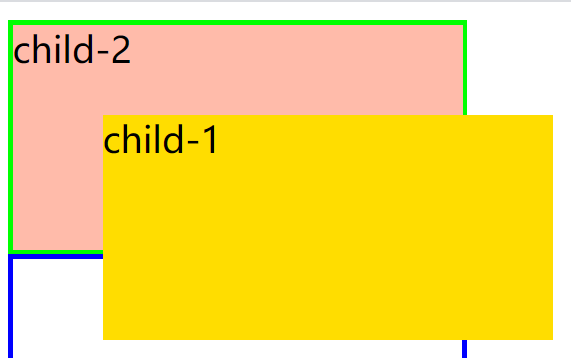


1. **relative**：生成相对定位的元素，**相对于元素本身在normal flow文档流中的正常位置进行定位**。**不脱离文本流**，**原有的位置还是占据着**。如果不设置TLBR(top,left,bottom,right)的话，它的位置不会被改变，且不会影响当前布局，相当于没事发生一样。如果设置了TLBR后，元素就可以向指定的方向偏移。例子如图：（不脱离文档流）

对child1设置：position: relative; left: 50px; top: 50px;



1. **absolute**：生成绝对定位的元素，**相对于 static 定位以外的第一个父元素进行定位**，元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。**如果元素的父级都没有设置定位属性，absolute元素将以body为父级，相对于body坐标原点进行定位**。（**脱离文档流，原来的位置不再占有**）



（可以看到父级box的高度都变小了，child-2完全顶替了child-1的位置）

1. fixed：生成固定定位的元素，相对于**浏览器窗口**进行定位，不会随着页面滚动而滚动。

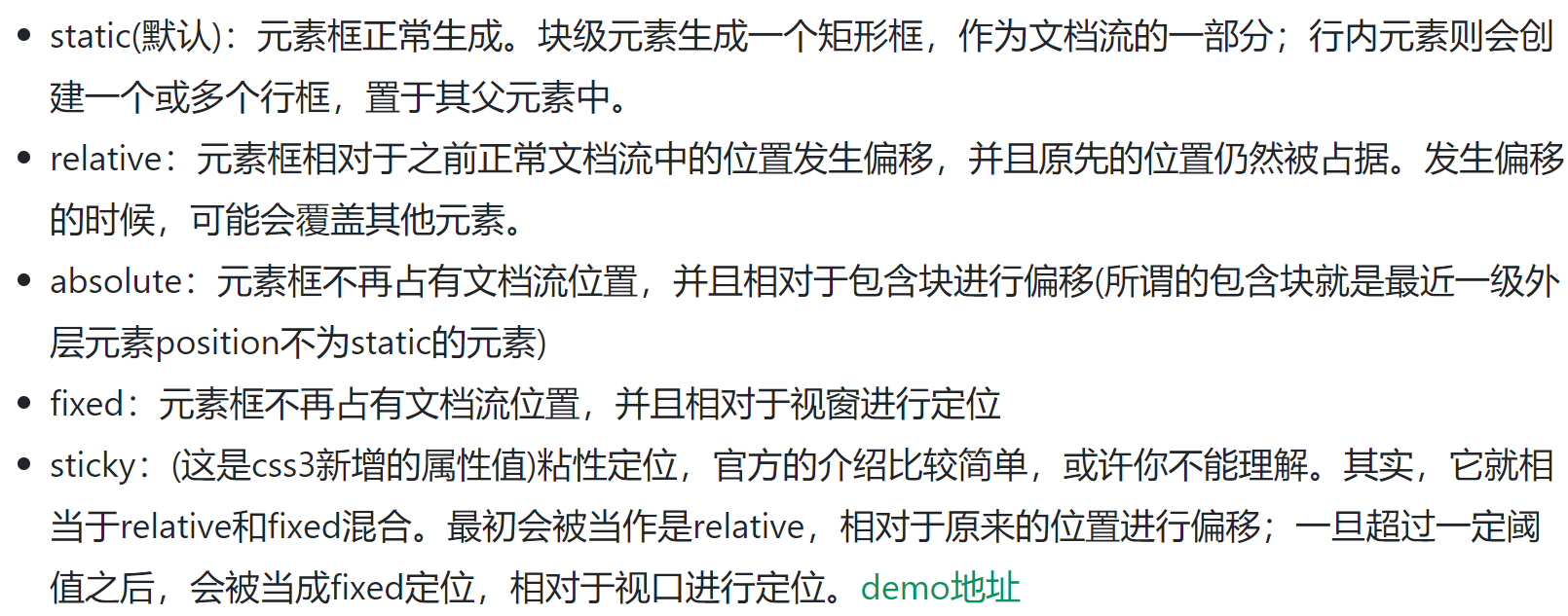
**脱离文档流**，这里所固定的参照对像是**可视窗口**而并非是body或是父级元素。使用了fixed的元素不会随着窗口的滚动而滚动。属于absolute的子集。

1. inherit：规定应该从父元素继承 position 属性的值。

relative定位的元素，**是相对于元素本身的正常位置来进行定位的**。

absolute定位的元素，是**相对于它的第一个position值不为static的父元素**的paddingbox来**进行定位**的。这句话我们可以这样来理解，我们首先需要找到绝对定位元素的一个position的值不为static的父元素，然后相对于这个父元素的paddingbox的**左上角**来定位，也就是说在计算定位距离的时候**，padding的值也要算进去**。

（top属性：定义了一个定位元素的上外边距边界与其**包含块**上边界之间的偏移。）



<https://segmentfault.com/a/1190000011358507>

## transform属性

transform属性应用于元素的2D或3D转换，可以对元素进行旋转、缩放、倾斜和移动。

旋转通过设置rotate来实现，在括号中设置旋转的度数。参数为正时，顺时针旋转；参数为负时，逆时针旋转。还可以分别设置rotateX()和rotateY()来只进行横轴或纵轴上的旋转。



缩放通过设置**scale**来实现，在括号中设置缩放倍率。当只设置一个参数时，横轴和纵轴方向缩放倍数相同，设置两个参数时第一个表示横轴缩放的倍数第二个表示纵轴缩放的倍数。也可以设置scaleX()：来只在横轴方向上进行缩放，纵轴方向不变；设置scaleY()只在纵轴方向上进行缩放，横轴方向不变。

倾斜通过设置**skew**来实现，如果只传一个参数，表示只沿横轴变换；如果传两个参数，第一个沿横轴变换，第二个沿纵轴变换。同样也可以分别设置skewX()和skewY()来只进行沿横轴倾斜变换或沿纵轴倾斜变换

移动通过设置**translate**来实现，传递一个参数时，表示只在横轴方向上移动；两个参数时，第一个参数是横轴移动的距离，第二个参数是纵轴移动的距离，（当距离为正的时候移动的方向是向右和向下，距离为负时向左和向上。）同样也可分别设置translateX()和translateY()来只在横轴方向或纵轴方向进行平移。

进行以上这些变换的时候，是以**元素的中心点**为基准点进行变换的。使用transform-origin属性可以改变变换的基准点。如通过设置transform-origin：left bottom可以将变换的基准点设置为元素的左下角。

## flex弹性/伸缩盒模型？

flex是CSS3新增的一种布局方式，我们可以通过将一个元素的display属性值设置为flex从而使它成为一个flex容器，它的所有子元素都会自动成为容器成员，称为项目。

一个flex容器默认存在两条轴：水平的主轴和垂直的交叉轴。我们可以使用flex-direction来指定主轴的方向，即项目的排列方向。使用justify-content来定义项目在主轴上的对齐方式，使用align-items来定义项目在交叉轴上的对齐方式。还可以使用flex-wrap来规定当项目在一条轴线上排列不下时的换行方式。

对于容器中的项目，我们可以使用order属性来指定项目的排列顺序，使用flex-grow来定义项目的放大比例，使用flex-shrink来定义项目的缩小比例。

## 水平垂直居中方式、手写实现（作业帮、美团）

<div class = “a”>

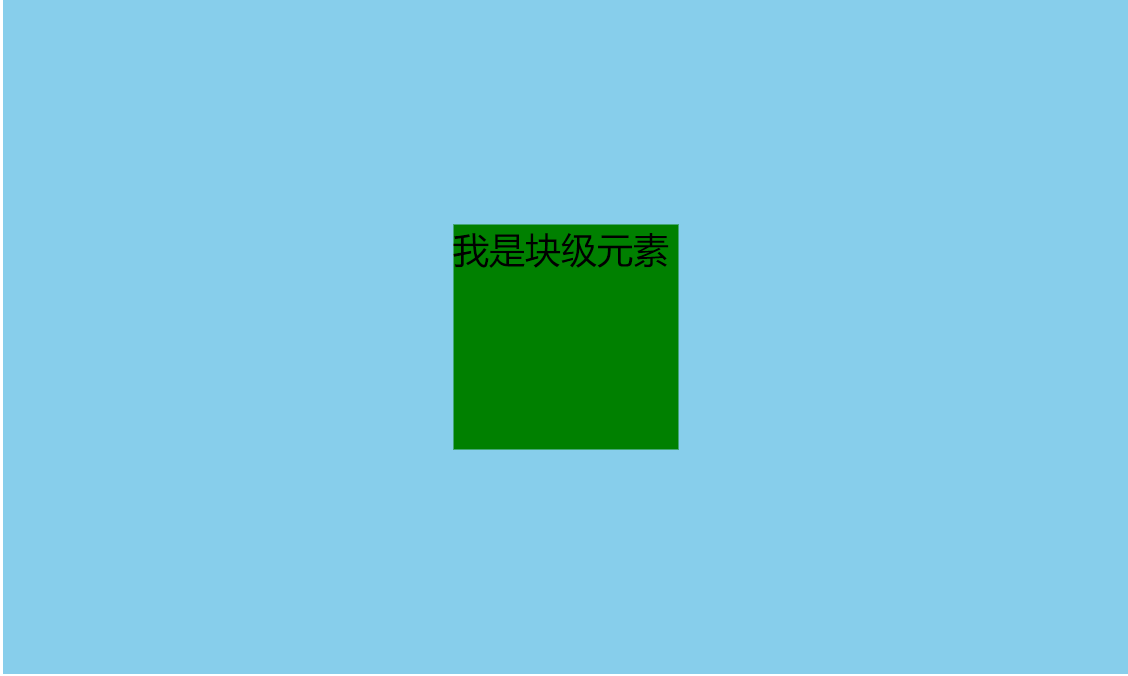
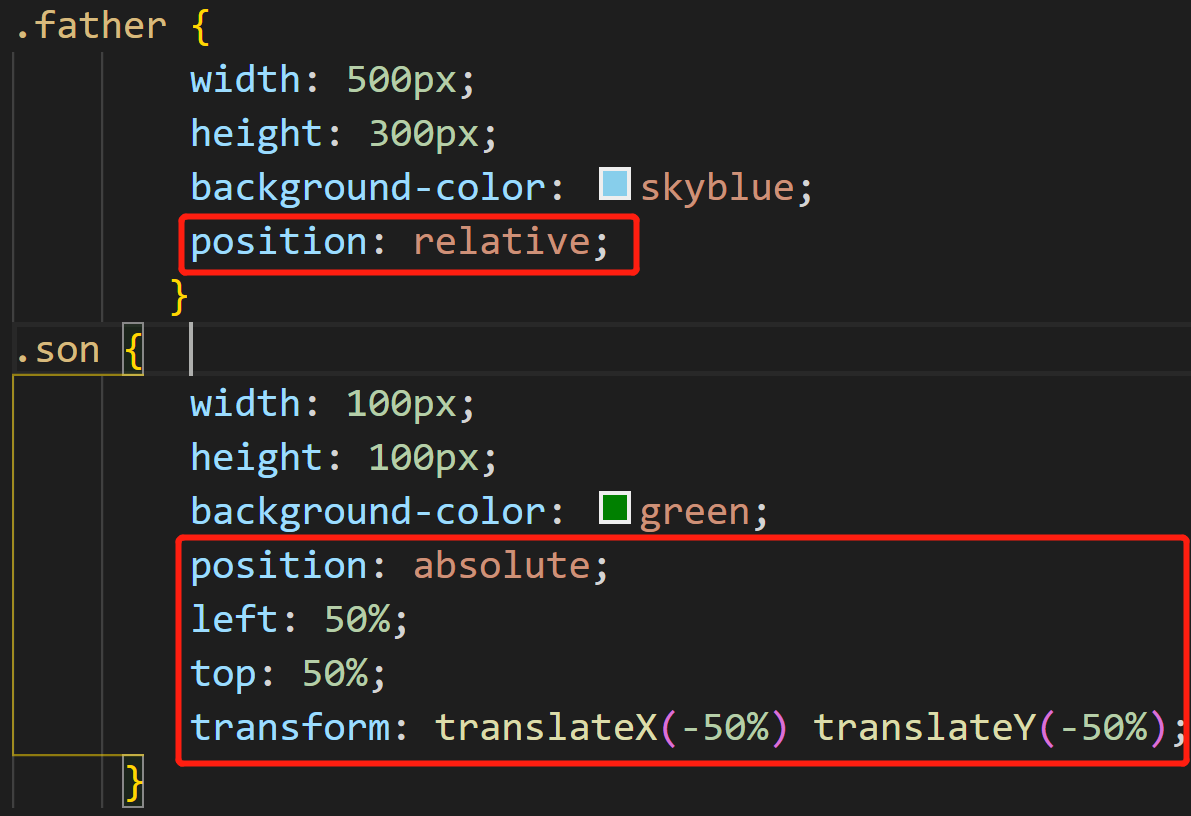
<div class = “b”></div>

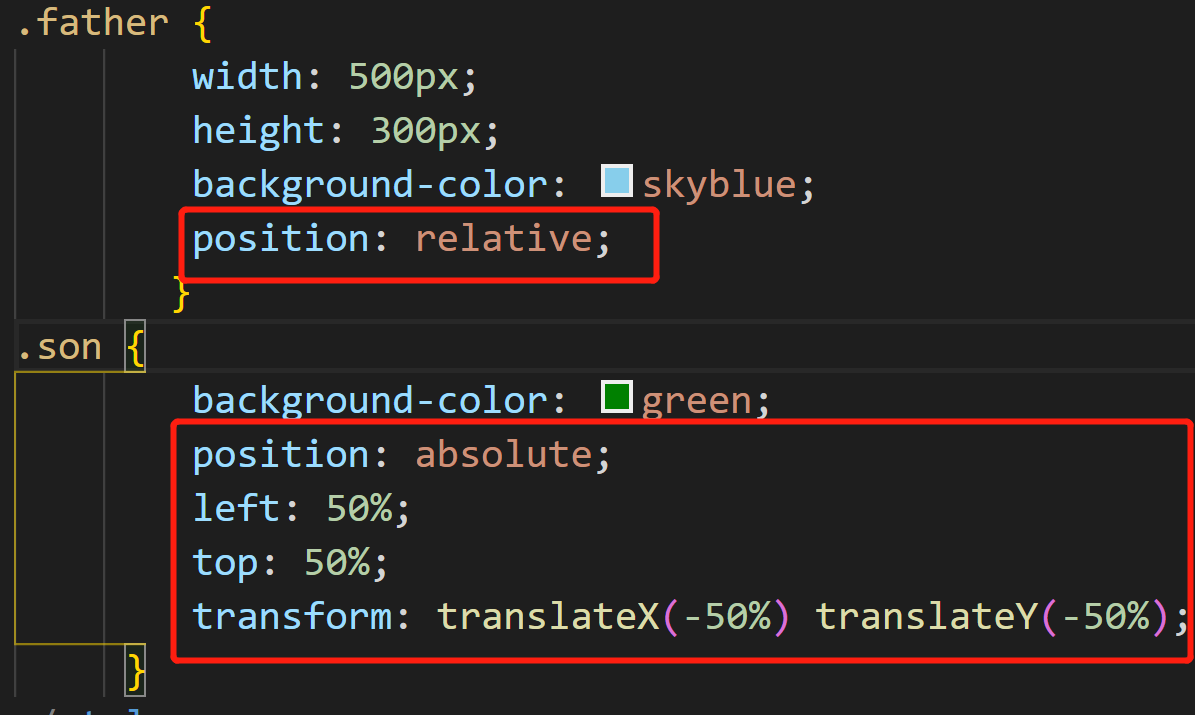
</div>



### 方案一：使用定位属性

使用定位属性，设置父元素为相对定位，给子元素设置绝对定位**，left: 50%; top: 50%; transform: translateX(-50%) translateY(-50%);**

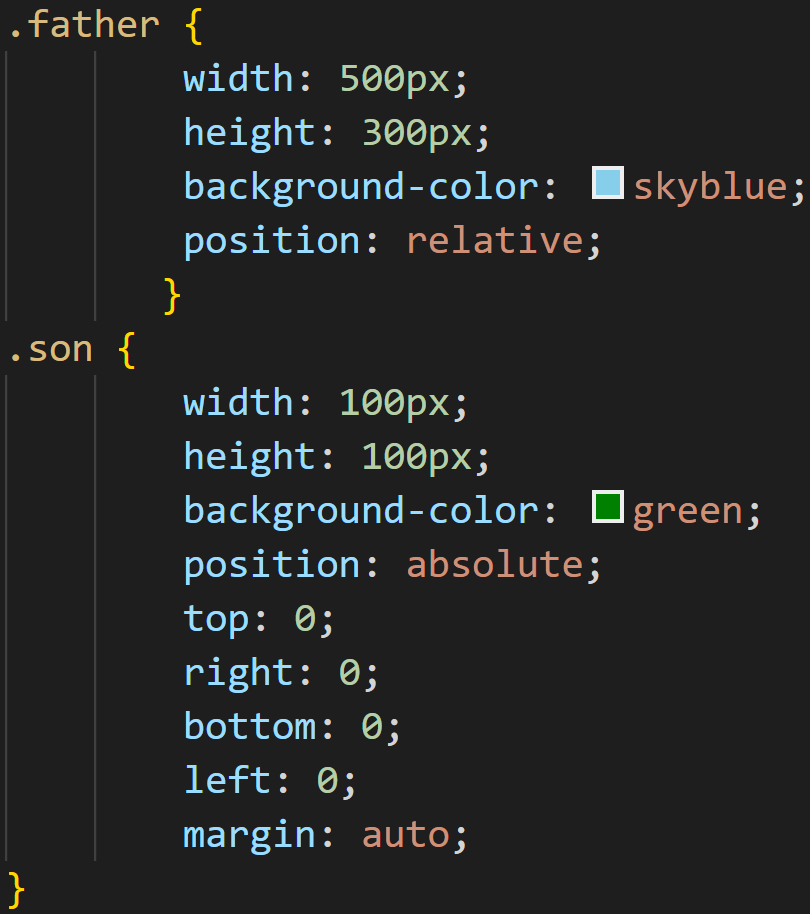
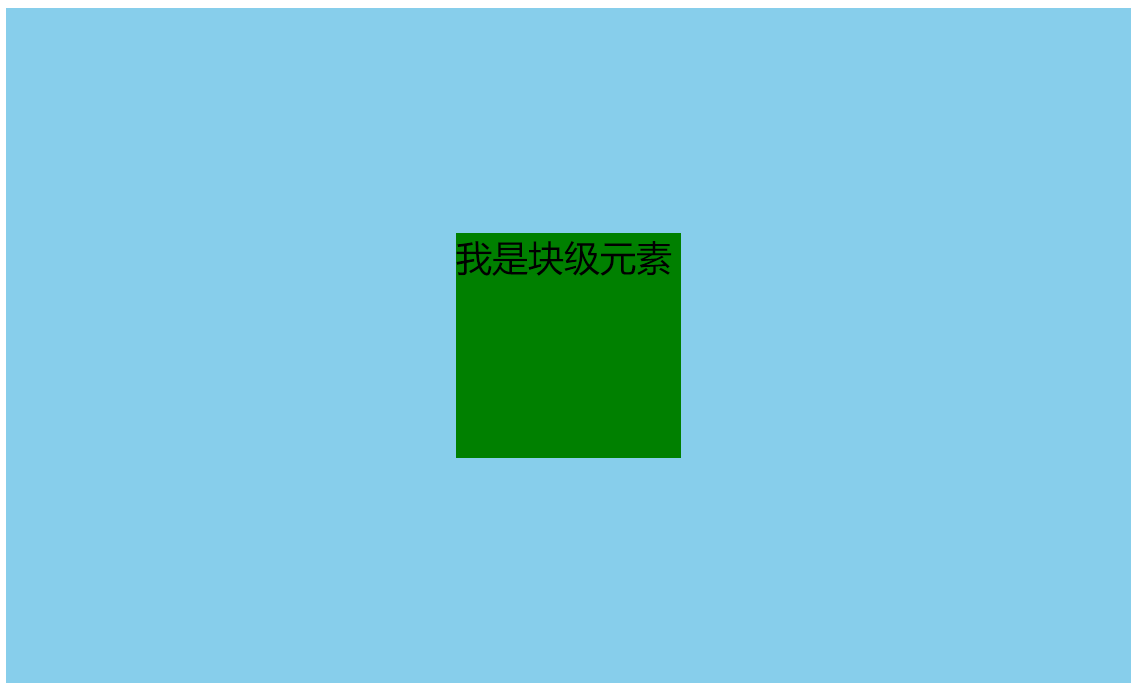




若已知高度和宽度：



1. 设置父元素为相对定位，给子元素设置绝对定位，**left: 0; top: 0; right: 0; bottom: 0; margin: auto;**

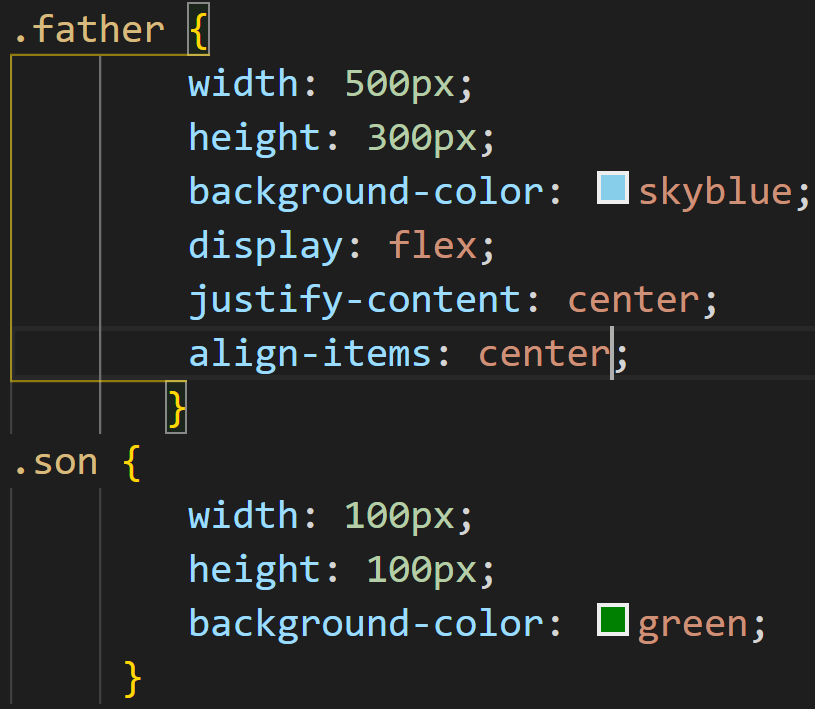
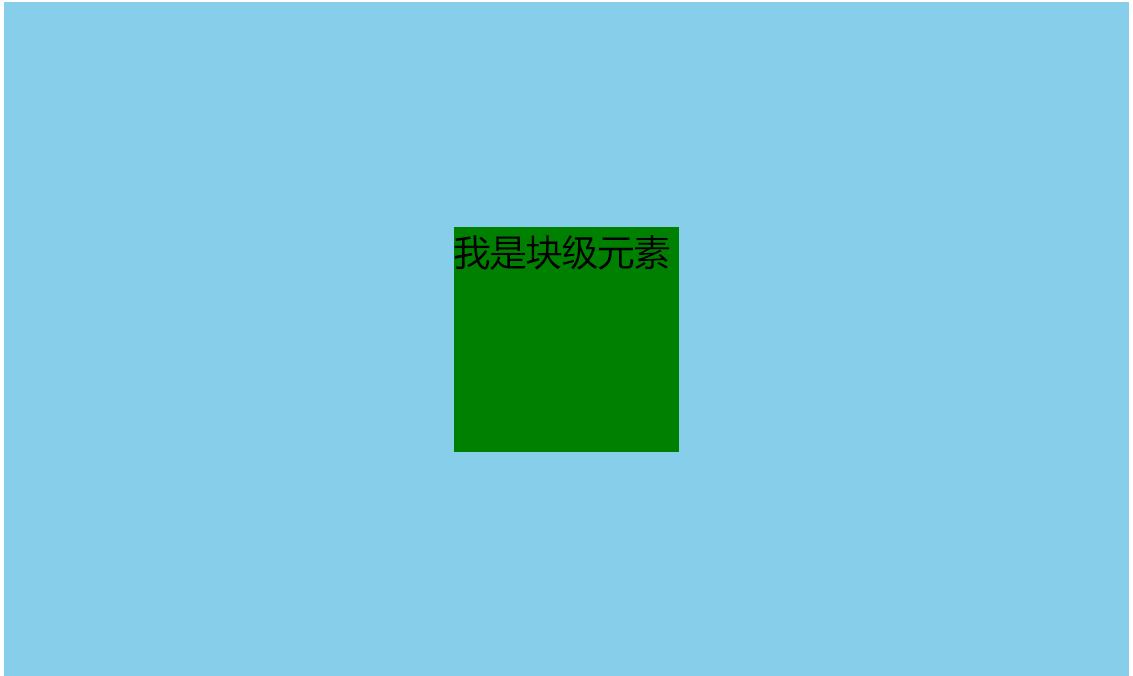
1. 方法二：设置父元素为相对定位，给子元素设置绝对定位**，left: 50%; top: 50%; margin-left: -元素宽度的一半px; margin-top: -元素高度的一半px;**

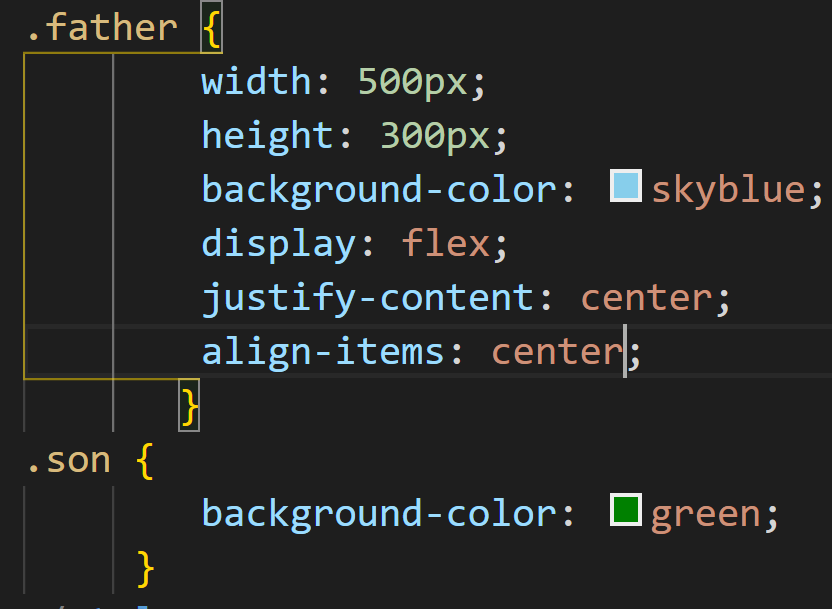
效果同上

### 方案二：使用flex布局实现

**设置父元素为flex布局，justify-content: center; align-items: center;**（无论是否已知宽高都可以的）

效果同上

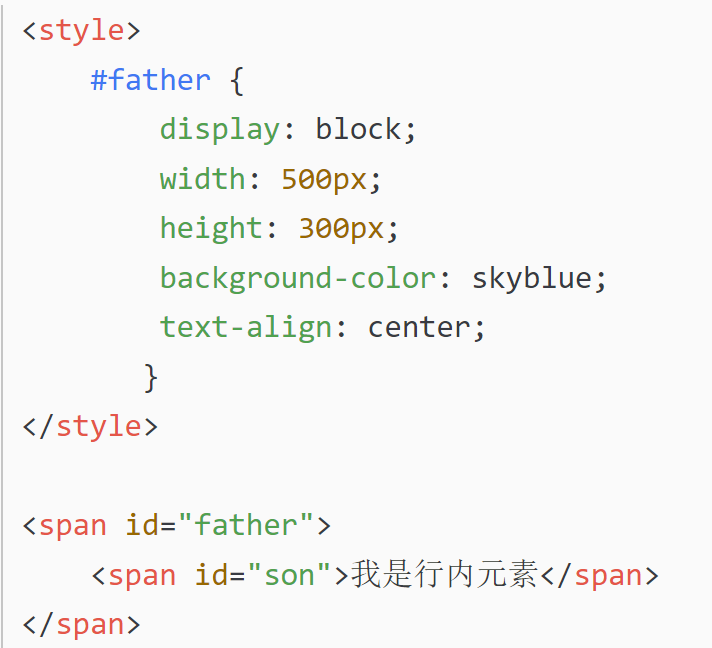
### 水平居中的方式：

**行内元素**：

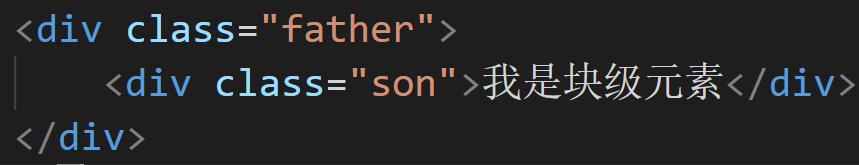
首先看它的父元素是不是块级元素，如果是，则**直接给父元素设置 text-align: center;**



如果不是，则先将其父元素设置为块级元素，再给**父元素设置 text-align: center**;

效果同上

**块级元素**：

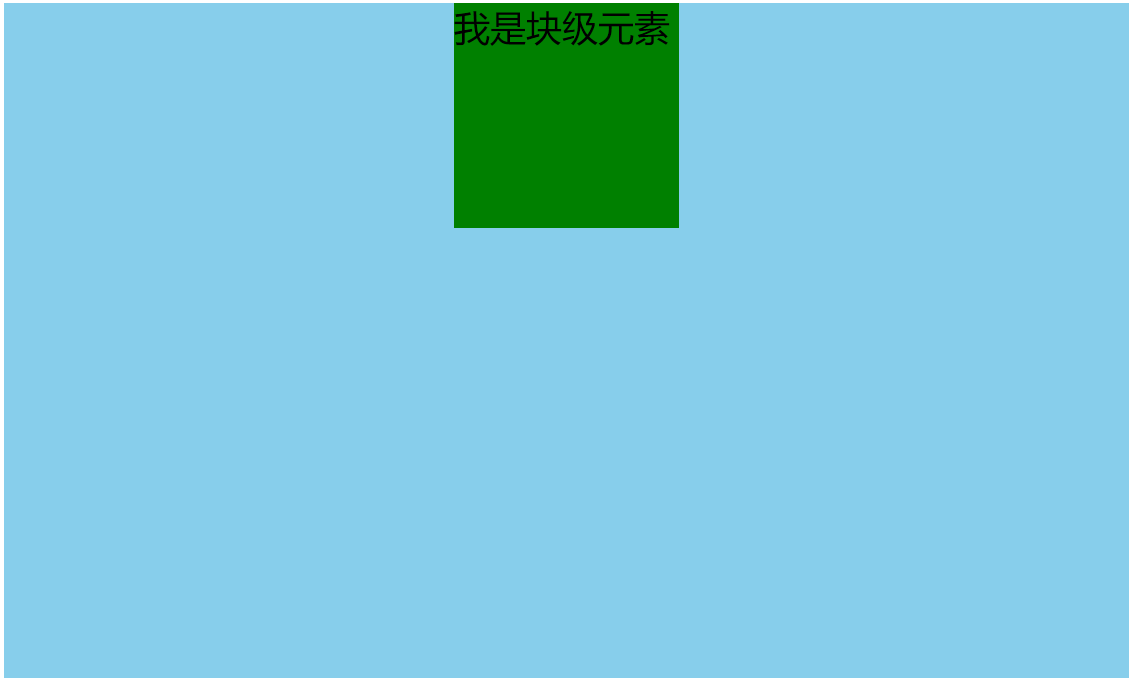
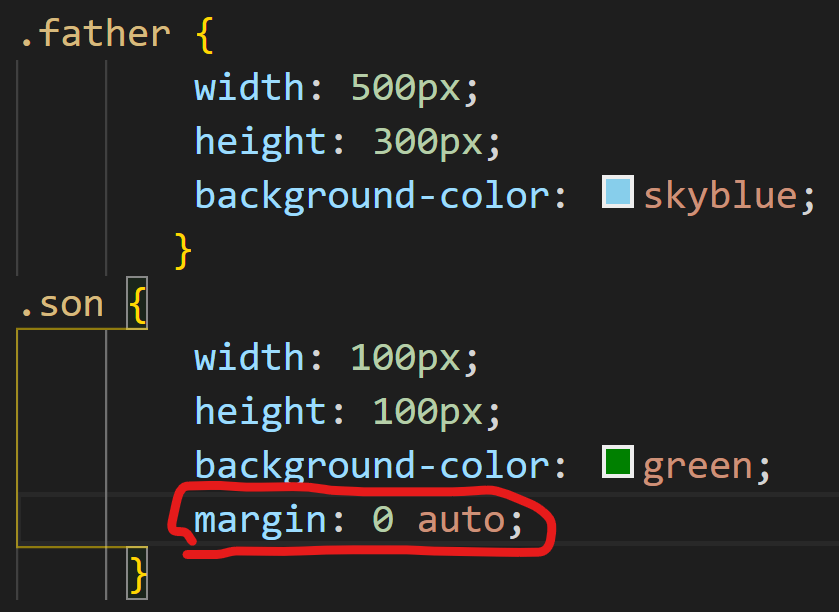


### 方案一：传统方法，需要分定宽和不定宽两种情况，

1. 定宽：原始



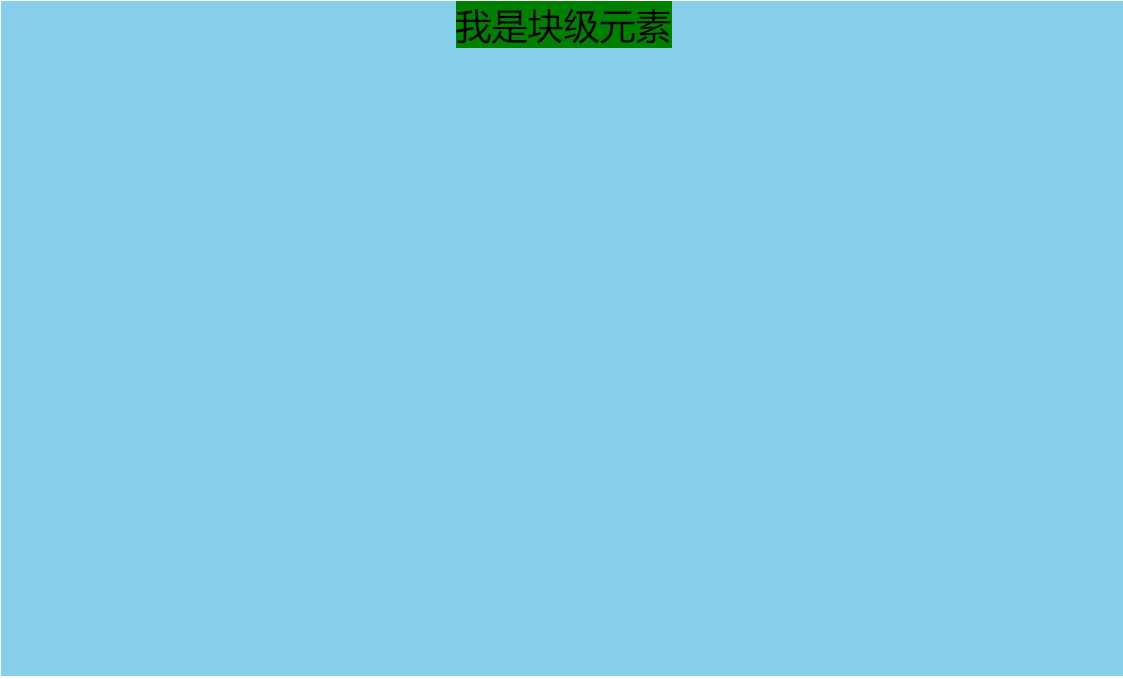
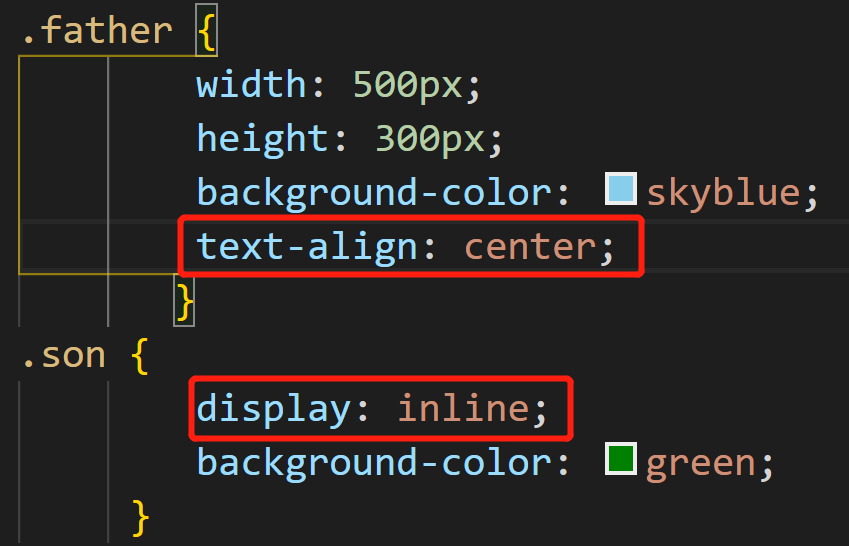
需要谁居中，**给其设置 margin: 0 auto;** （作用：使盒子自己居中）



1. 不定宽：默认子元素的宽度和父元素一样



需要设置子元素为display: inline-block; 或 display: inline;即将其转换成行内块级/行内元素，给父元素设置 text-align: center;



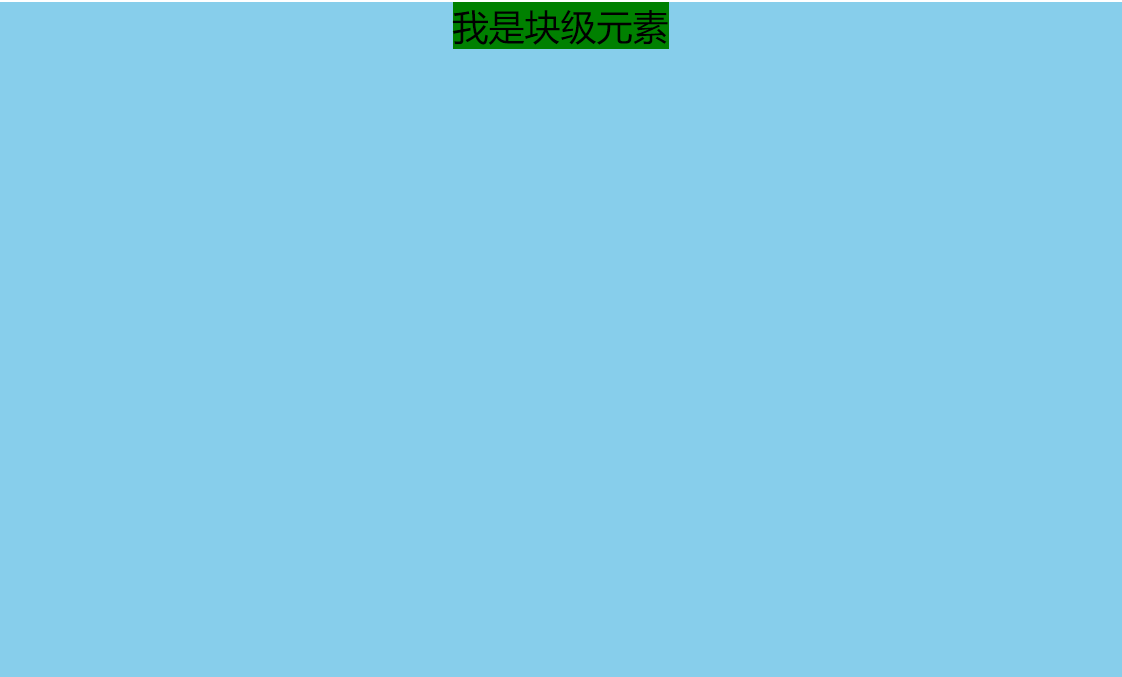
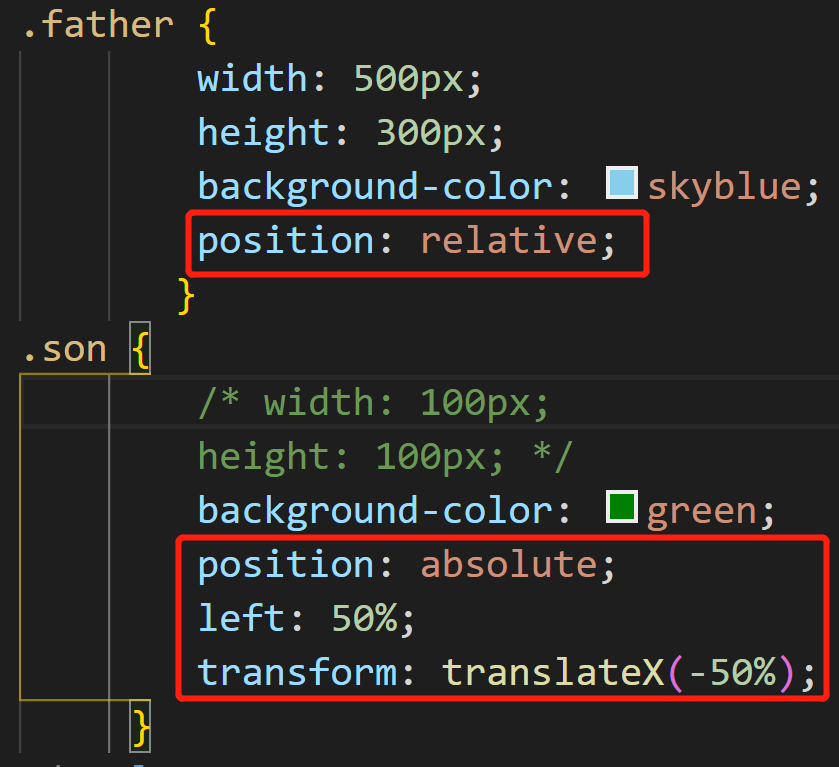
### 方案二：使用定位属性

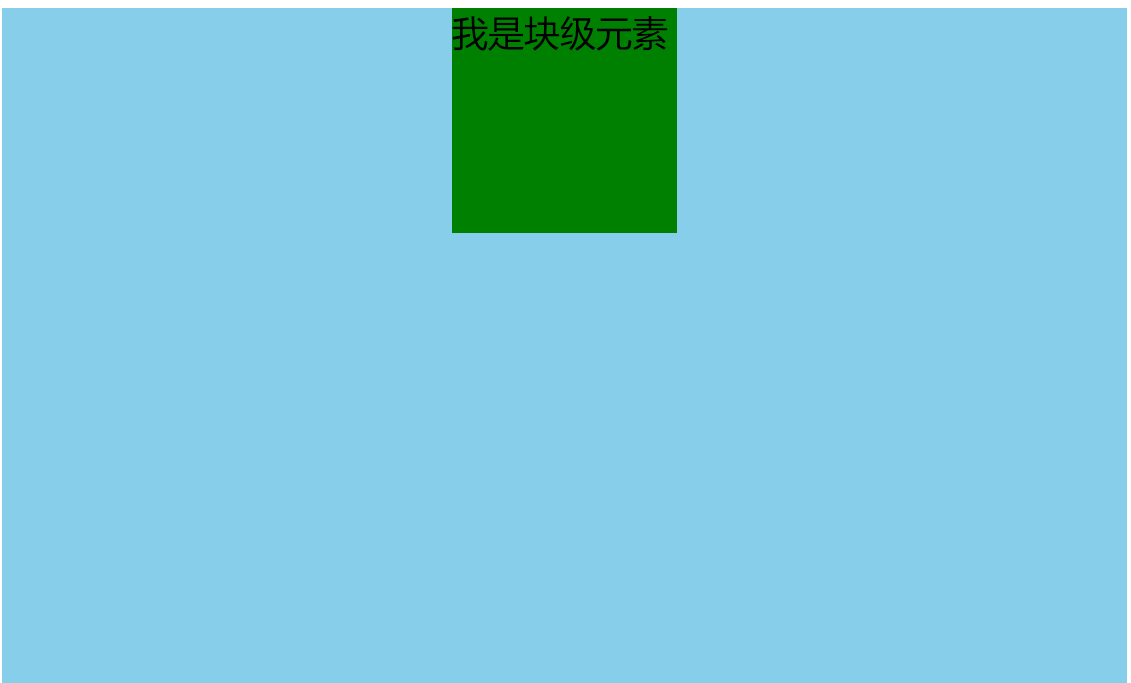
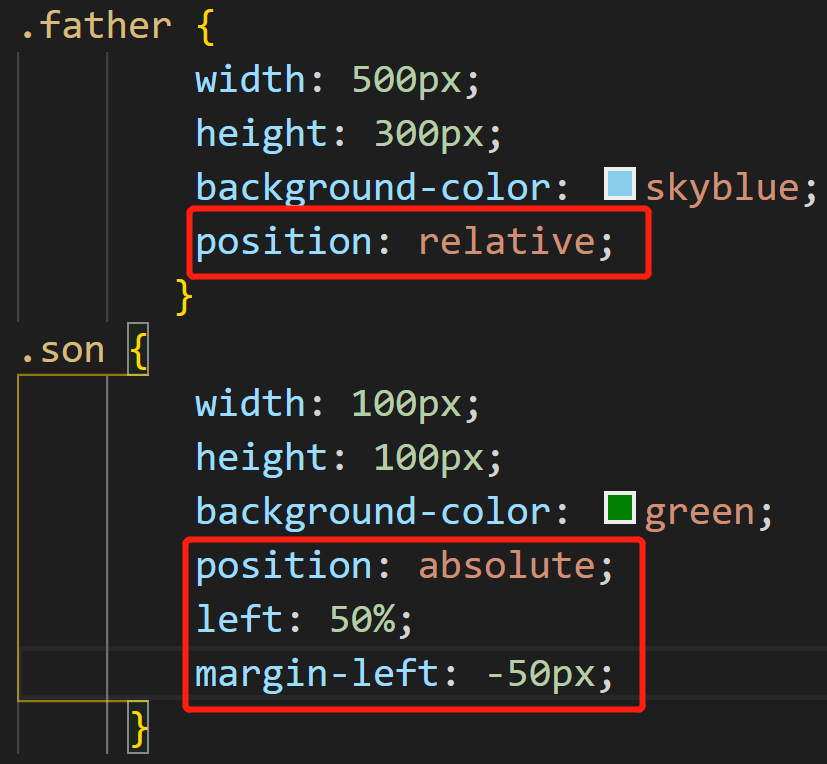
**首先设置父元素为相对定位，再设置子元素为绝对定位，设置子元素的left:50%**，即让子元素的**左上角**水平居中；

1. 设置transform: translateX(-50%);（无论是否定宽）

定宽时的效果同②

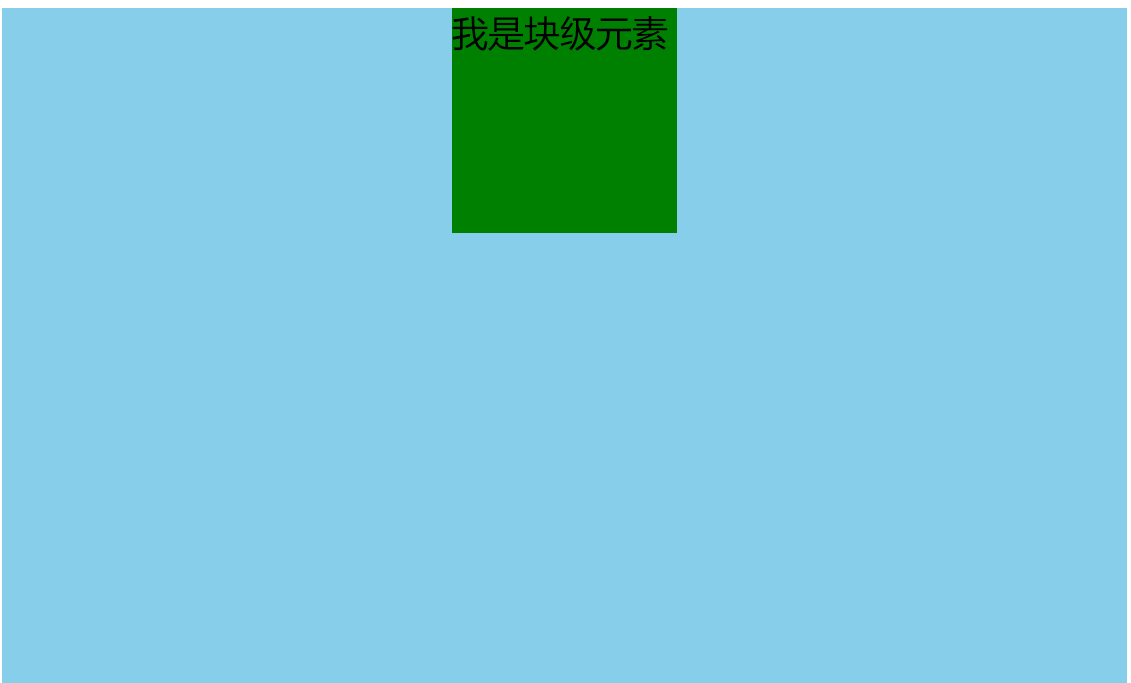
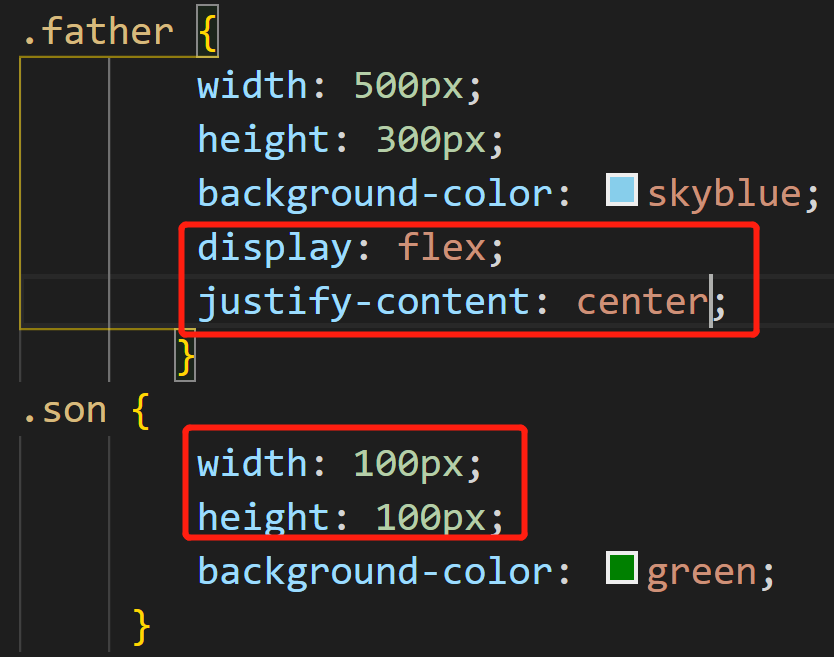
不定宽则：

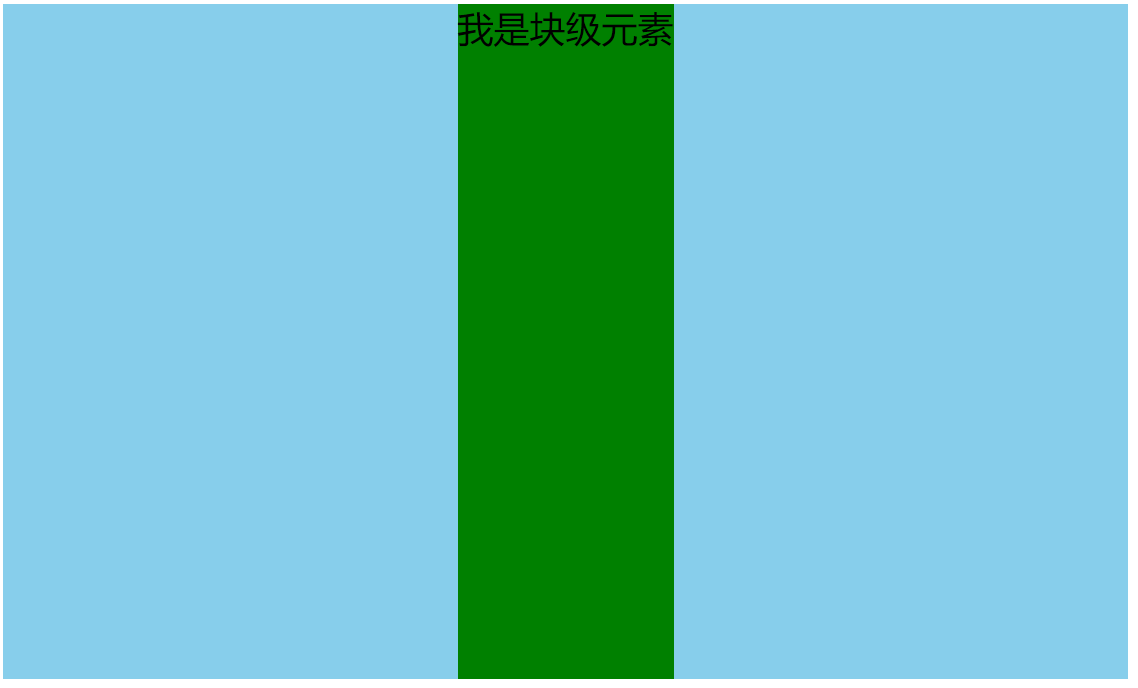
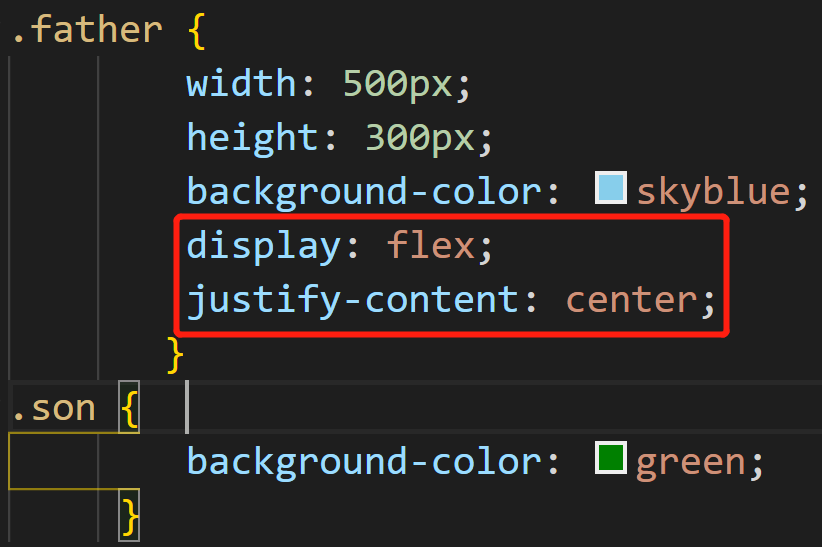


1. 定宽：设置**绝对子元素的margin-left: -元素宽度的一半px**

### 方案三：使用flex布局实现

使用flex布局，只需要给待处理的块状元素的父元素添加属性 display: flex; justify-content: center;无论宽度定不定：





## CSS有哪些选择器？优先级？（作业帮）

CSS的选择器有：

标签选择器（body{ }、p{ }）

类选择器（.first{ }）

id选择器(#first{ })

属性选择器（[id=section]{ }）

伪类选择器（a:hover{ }）

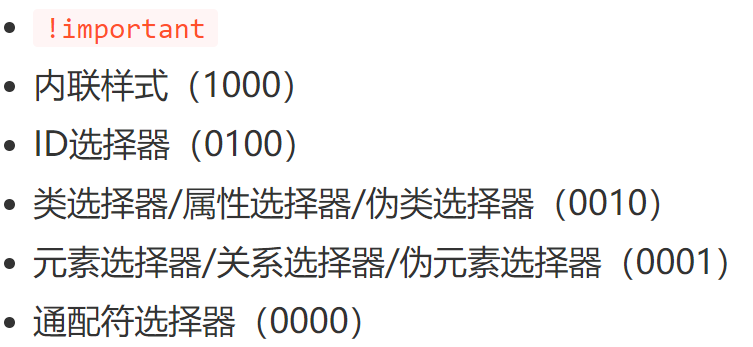
伪元素选择器（p::before{ }）

通用选择器/全局选择器（\*{ }）

（还有子选择器、后代选择器、分组选择器等）

优先级取决于选择器的特殊性，特殊性越高，优先级越高。选择器的特殊性可以分为四个等级，第一个等级是行内样式，特殊性值为1000，第二个等级是id选择器，为0100，第三个等级是类选择器、伪类选择器和属性选择器，为0010，第四个等级是元素选择器和伪元素选择器，为0001。**样式中每出现一个选择器，就将它的特殊性进行叠加，**（这个叠加只限于对应的等级的叠加，不会产生进位。）**选择器特殊性值的比较是从左向右排序的**，也就是说以1开头的特殊性值比所有以0开头的特殊性值要大。比如说特殊性值为1000的的规则优先级就要比特殊性值为0999的规则高。**如果两个规则的特殊性值相等的时候，那么就会根据它们引入的顺序**，**后出现的优先级更高**。

不同级别时：



**不同级别：！important（在样式属性后不加“;”，在!important后加分号）>内联/行内样式>ID选择器>类选择器>标签>通配符**

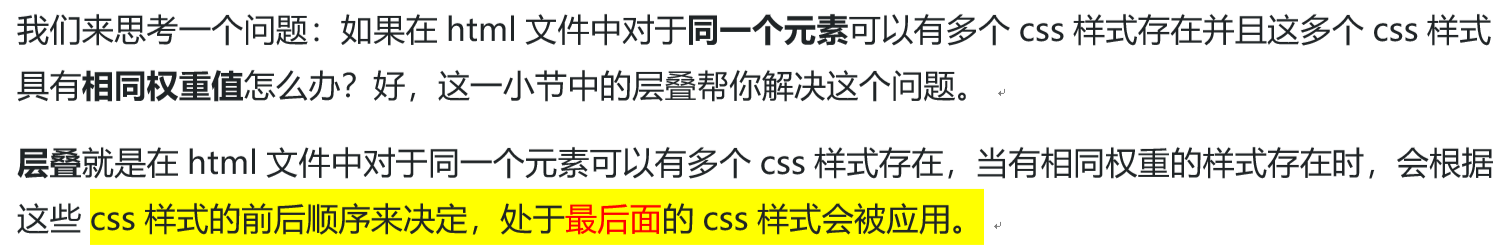
**同一级别：后写的会覆盖先写的样式**

### 七牛云的问题：

两个外部css文件对页面中同一个html元素的背景色进行设置，一个设置为绿色，一个设置为红色，那么这个html元素的背景色到底是红色还是绿色？

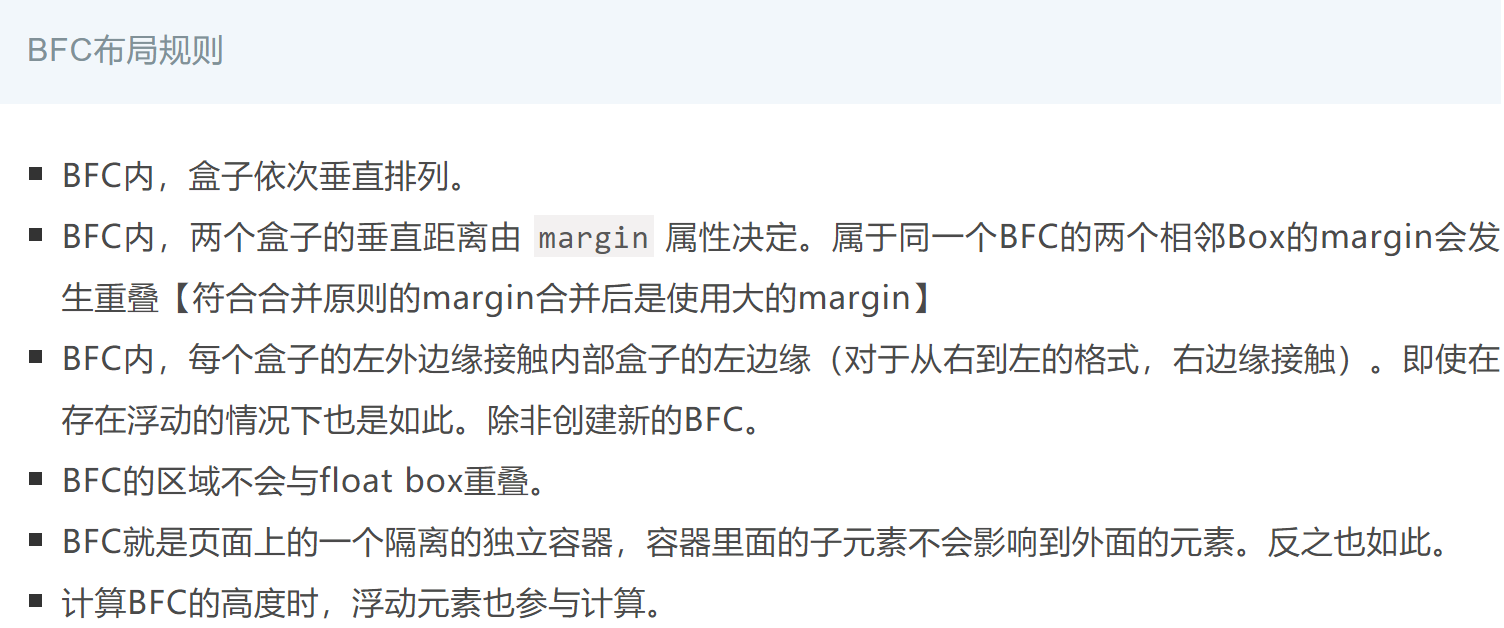
答：**由引入CSS文件的顺序决定，后引入的样式会覆盖先引入的样式**，见笔记中*选择器优先级.html*的测试。（并且发现采用**嵌入式**的方式时，也是取决于定义样式时的顺序，后定义的覆盖先定义的）

其实就是在问“层叠”

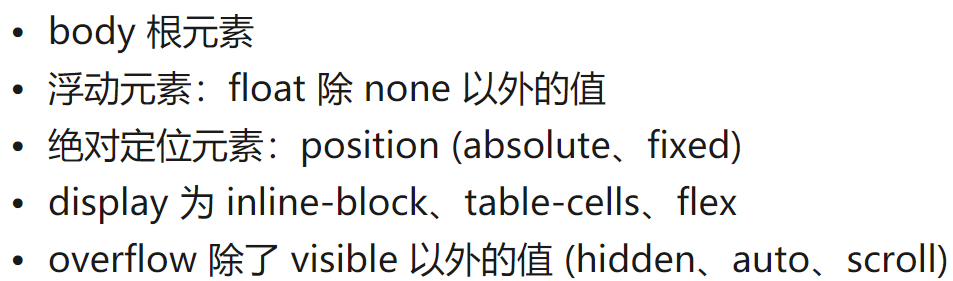


## BFC？BFC的布局规则是什么？如何创建BFC？

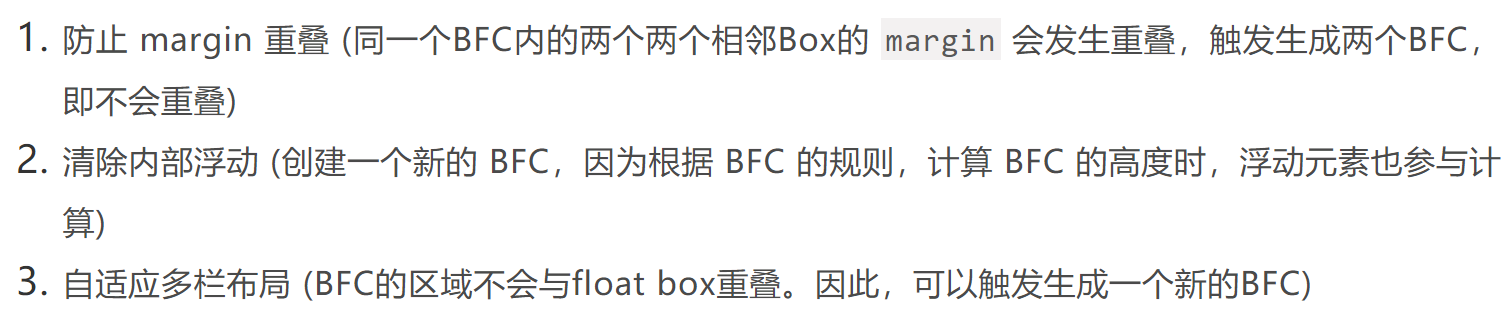
定义：BFC即块级格式化上下文，是页面中的一块渲染区域。一个元素形成了BFC之后，那么它内部元素产生的布局不会影响到外部元素，外部元素的布局也不会影响到BFC中的内部元素。通俗来说，一个BFC就像是一个隔离区域，和其他区域互不影响。



只要元素满足下面任一条件即可触发 BFC 特性/创建BFC元素：



BFC的应用：



3前加上可以**阻止元素被浮动元素覆盖，实现两列自适应布局**，效果不错，这时候左边的宽度固定，右边的内容自适应宽度(去掉上面右边内容的宽度)。

<https://zhuanlan.zhihu.com/p/25321647>

## 如何实现三栏布局，说几种方法（七牛云一面凉）

描述：三块排在同一行，每个块的宽度是33.3%（面试官期待听到flex布局的方式！）

## CSS画一个三角形（美团一面）

## html怎么给一个元素添加一个class

setAttribute() classList.add()（听面试官的语气应该还有别的方法）

## 用过html5的语义化标签吗？这些标签有什么优点？（携程一面）

用过，比如有<article>、<section>、<nav>、<aside>、<header>、<footer>等等。



所谓的语义化是指这些标签的**名称**本身传达了**关于这些标签所包含的内容**特点的一些信息。

优点：

1. 文档/页面**结构**更加清晰明确，比如文档的主要内容，导航栏还有标题、脚注等分工明确。
2. 使代码的**可读性**更好，便于团队开发和维护。
3. **方便其他设备解析**，如屏幕阅读器、pad、手机等移动设备，使这些设备以有意义的方式来渲染网页。

PDA、手机等设备可能无法像普通电脑的浏览器一样来渲染网页（通常是因为这些设备对CSS的支持较弱）。使用语义标记可以确保这些设备以一种有意义的方式来渲染网页.理想情况下,观看设备的任务是符合设备本身的条件来渲染网页.语义标记为设备提供了所需的相关信息,就省去了你自己去考虑所有可能的显示情况（包括现有的或者将来新的设备）.例如,一部手机可以选择使一段标记了标题的文字以粗体显示.而掌上电脑可能会以比较大的字体来显示.**无论哪种方式一旦你对文本标记为标题,您就可以确信读取设备将根据其自身的条件来合适地显示页面**.

1. **有利于SEO**(Search Engine Optimization，搜索引擎优化)：搜索引擎的爬虫依赖标签来确定关键字的权重，因此语义化标签可以和搜索引擎建立良好的沟通，帮助爬虫抓取更多的有效信息。

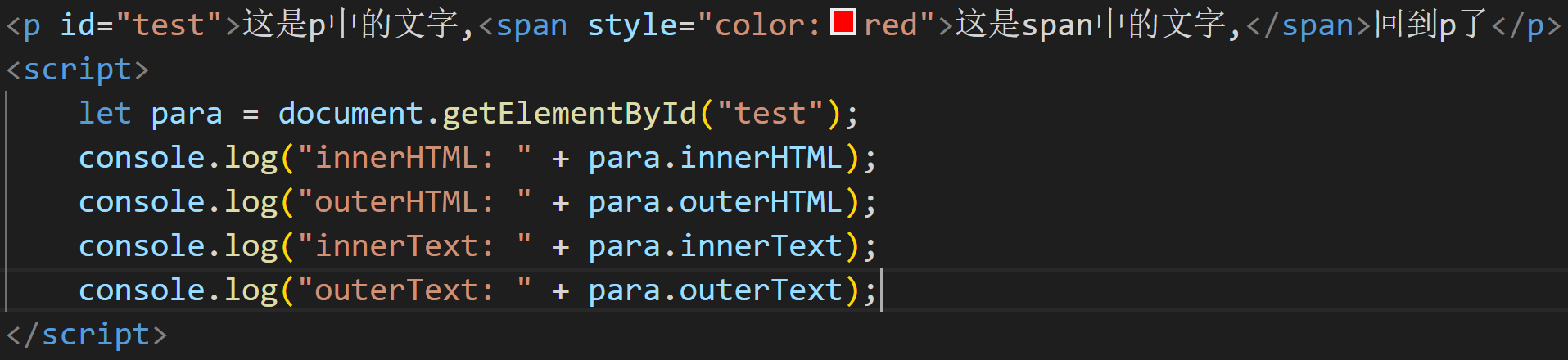
搜索引擎的爬虫也依赖于标记来确定上下文和各个关键字的权重.过去你可能还没有考虑搜索引擎的爬虫也是网站的“访客”,但现在它们他们实际上是极其宝贵的用户.没有他们的话,搜索引擎将无法索引你的网站,然后一般用户将很难过来访问

1. 有利于合作，减少差异化，因为都遵守W3C标准

## innerHTML和outerHTML、innerHtml和innerText的区别？(携程一面)

* innerHTML设置或获取**位于对象起始和结束标签内**的**HTML**
* outerHTML设置或获取对象及其内容的HTML形式，在innerHTML基础上还包含了对象标签本身。
* innerText设置或获取位于对象起始和结束标签内的**文本（只拿文本）**
* outerText设置(包括标签)或获取(不包括标签)对象的**文本**（不太重要）

实例：

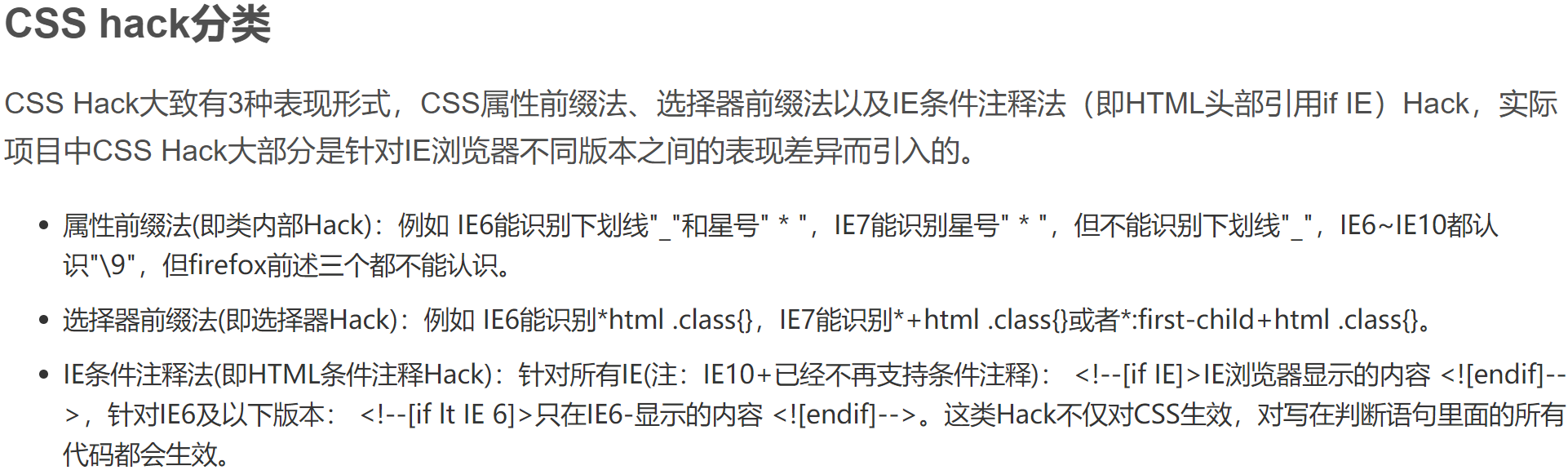




## css hack?（为了适配使用的一些非正常css写法？）（携程一面）

（感觉比较生僻，先简单整理以下概念）

概念：由于不同的浏览器（如IE、Chrome、Firefox等或同一浏览器的不同版本如IE6/7/8）对CSS的支持和解析不完全一样，因此会导致在不同浏览器的环境中生成的页面效果不一样，这时，为了获得统一的页面效果，我们就需要针对不同的浏览器，去写不同CSS代码，让它能够同时**兼容不同的浏览器。我们把这个针对不同的浏览器/不同版本写相应的CSS code的过程，叫做CSS hack!**







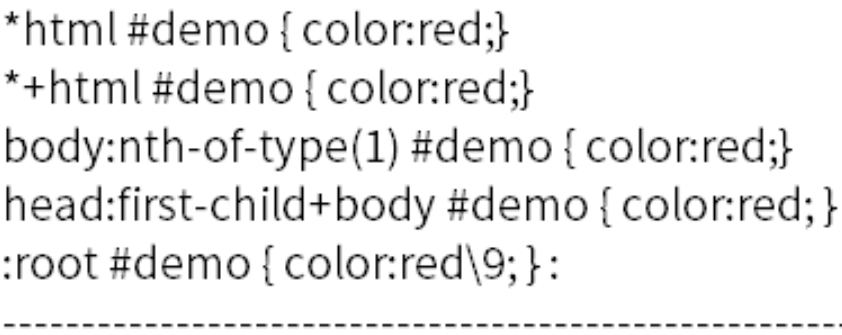




还有在属性值后面添加“!important”的写法只有IE6不能识别，其它版本IE及现代浏览器都可以识别,

总结：



CSS hack利弊：

CSS hack是因为现有浏览器对标准的解析不同，为了兼容各浏览器，所采用的一种补救方法。CSS hack是一种类似作弊的手段，以欺骗浏览器的方式达到兼容的目的，是用浏览器的兼容性差异来解决浏览器的兼容性问题。一般情况下，我们尽量避免使用CSS hack，但是有些情况为了顾及用户体验实现向下兼容，不得已才使用hack。比如由于IE8及以下版本不支持CSS3,而我们的项目页面使用了大量CSS3新属性在IE9/Firefox/Chrome下正常渲染，这种情况下如果不使用css3pie或htc或条件注释等方法时,可能就得让IE8-的专属hack出马了。使用hack虽然对页面表现的一致性有好处，但过多的滥用会造成**html文档混乱不堪，增加管理和维护的负担**。

## 占位