前端开发核心知识进阶：50 讲从夯实基础到突破瓶颈
来自 Lucas ... · 盐选专栏

查看详情 >

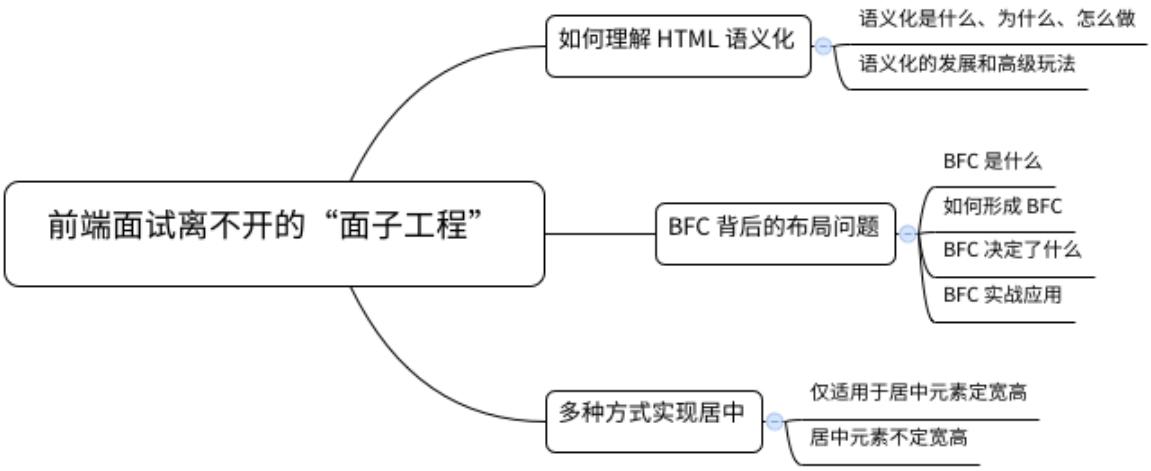
前端面试离不开的「面子工程」

我们都知道前端开发中的「三驾马车」：HTML + CSS + JavaScript。从难易程度、受关注程度上来讲，显然 JavaScript 始终处于核心地位。但是这并不意味着 HTML 和 CSS 不重要，如果你轻视它们，那么也许会在工作开发中、甚至面试中吃亏。

作为多年的面试官，我的考察重点无疑是 JavaScript，但在面试过程中，每次也总是会「蜻蜓点水」下，这足以了解候选者对待 HTML 和 CSS 的态度以及了解程度。其实事实上，HTML 和 CSS 也有很多有趣的内容，下面就让我们在复习重点知识的同时，了解一些前沿用法。

这一讲，我挑选出了 HTML 和 CSS 几个关键概念，不去力求「面面俱到」，但希望给大家带来新的启发。

相关知识点如下：



如何理解 HTML 语义化

HTML 语义化——这个概念其实诞生了挺长时间，我经常发现在面试 JD（Job Description） 中出现要求候选者「了解 HTML 语义化」、「对 HTML 语义化有深刻认知」的需求。对于这么一句 JD 范式标配，如果面试官真的问起，该如何回答呢?

语义化是什么、为什么、怎么做

简单来说，HTML 语义化就是：

根据结构化的内容，选择合适的标签。

那么为什么要做到语义化呢?

直观上很好理解，「合适的标签」是内容表达的高度概括，这样浏览器爬虫或者任何机器在读取 HTML 时，都能更好地理解，进而解析效率更高。这样带来的收益如下：

有利于 SEO

开发维护体验更好

用户体验更好（如使用 alt 标签用于解释图片信息）

更好的 accessibility，方便任何设备解析（如盲人阅读器）

那么如何做到语义化呢?

其实很简单，这就要求我们实时跟进、学习并使用语义化标签。这里我帮大家总结了一些典型的 HTML 标签，并进行分类。

Head	Sections	Grouping	Tables	Forms	Interactive	Edits	Embedded	Text-level
doctype	body	p	table	form	details	del	img	a
html	article	hr	caption	fieldset	summary	ins	iframe	em
head	nav	pre	thead	legend	command		embed	strong
title	aside	blockquote	tbody	label	menu		object	i, b
base	section	ol, ul	tfoot	input			param	u, s, small
link	header	li	tr	button			video	abbr
meta	footer	dl, dt, dd	th	select			audio	q
style	h1 - h6	figure	td	textarea			source	cite
script	main	figcaption	col	option			canvas	dfn
noscript	address	div	colgroup	progress.....			area, map, track.....	sub, sup, code, br, var, span .....

我将 HTML 标签分为 9 大类别，每一种类别都包含有语义化的标签内容，小图如下：

Head	Sections	Grouping	Tables
doctype	body	p	table
html	article	hr	caption
head	nav	pre	thead
title	aside	blockquote	tbody
base	section	ol, ul	tfoot
link	header	li	tr
meta	footer	dl, dt, dd	th
style	h1 - h6	figure	td
script	main	figcaption	col
noscript	address	div	colgroup

Forms	Interactive	Edits
form	details	del
fieldset	summary	ins
legend	command	
label	menu	
input		
button		
select		
textarea		
option		
progress……		

Embedded	Text-level
img	**a**
iframe	**em**
embed	**strong**
object	**i, b**
param	**u, s, small**
video	**abbr**
audio	**q**
source	**cite**
canvas	**dfn**
area, map, track……	**sub, sup, code, br, var, span ……**

了解了这些语义化的标签，我们就可以按照「适合内容与否」，进行使用。关于选取标准，我也简单总结了一下，抽象成代码表达为：

```
if (导航) {
 return



}
else if (文稿内容、博客内容、评论内容...包含标题元素的内容) {
 return

}
else if (目录抽象、边栏、广告、批注) {
 return

}
else if (含有附录、图片、代码、图形) {
 return

                                }
                    else if (含有多个标题或内容的区块) {
                                return

                                }
                      else if (含有段落、语法意义) {
                                return
||

                                ||

                                ||

                               || ...

                                }

                               else {
```

```
return

}
```

语义化的发展和高级玩法

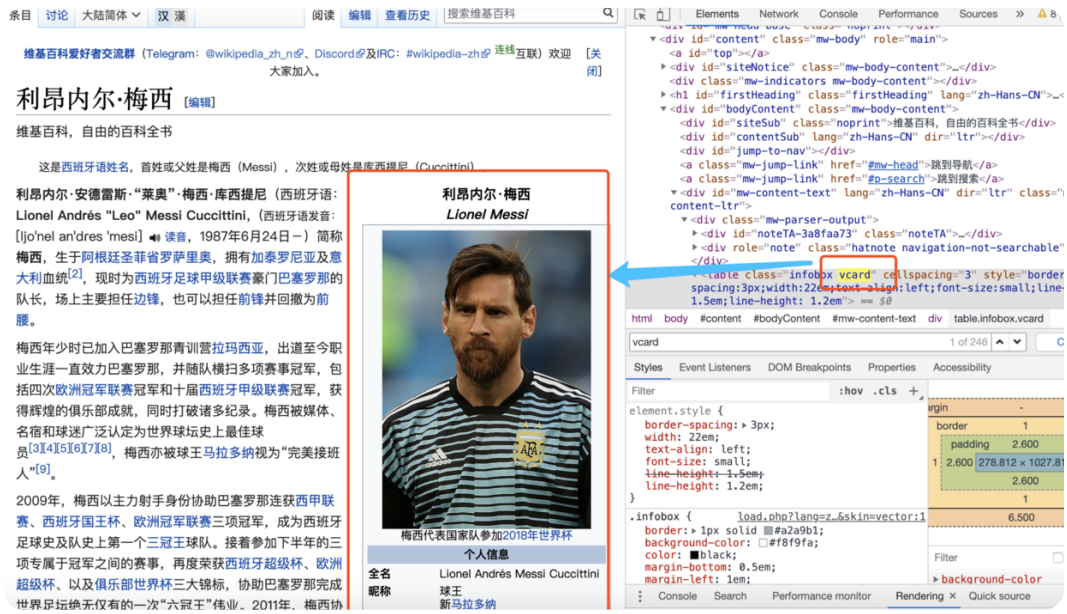
说到语义化的发展，我这里指向重点提一个概念：Microformats，如果面试官问的语义化时，你能把这个概念搬出来，效果是非常好的。那什么是 Microformats 呢?

Microformats，翻译为微格式，是 HTML 标记某些实体的小模式，这些实体包括人、组织、事件、地点、博客、产品、评论、简历、食谱等。它们是在 HTML 中嵌套语义的简单协议，且能迅速地提供一套可被搜索引擎、聚合器等其他工具使用的 API。

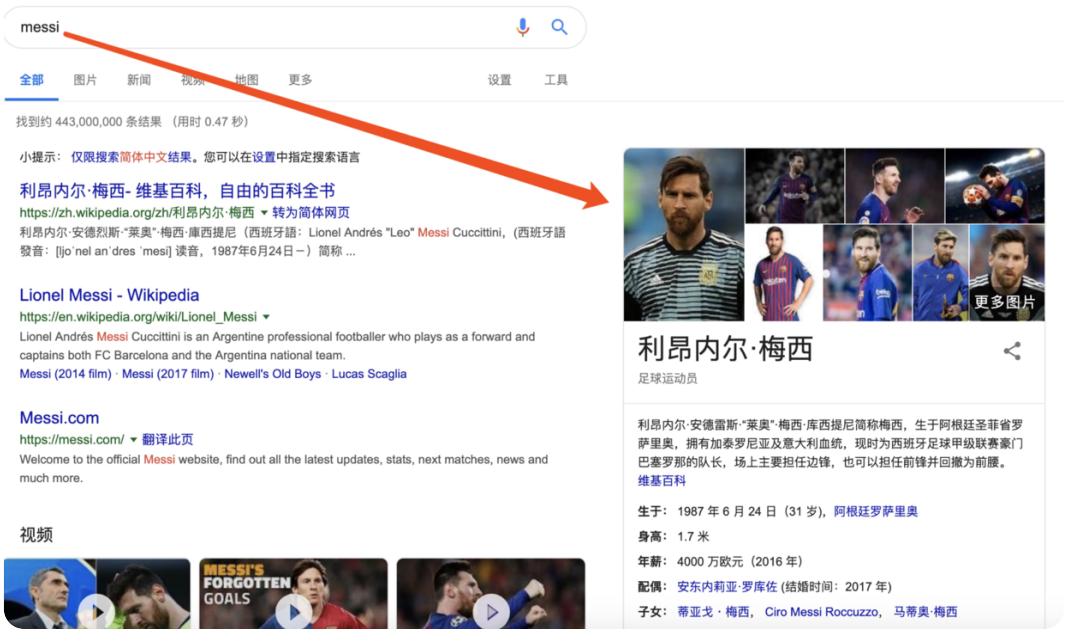
除了 hCard 和 hCalendar，有好几个库特别开发了微格式。

是不是看的一脸懵逼？其实很简单，Microformats 的原理就是扩展 HTML 元素或者属性，来增强 HTML 的语义表达能力。

我们来看一个案例：



Wikipedia 的页面中，给某一部分加上了 vCard 的 class，这是用来做什么的呢?



Google 搜索引擎可以通过 Wikipedia 页面 vCard 这个 class，读取相关内容，在呈现搜索结果时，匹配展现出人物信息。从而语义化的 class，帮助了机器（搜索爬出）学习到更多信息，展现出了更好的结果页面。

Microdata 属于 WHATWG（网页超文本应用技术工作小组：Web Hypertext Application Technology Working） HTML 规范，它并不是标准，但这是一个很典型的语义化发展和应用尝试。

BFC 背后的布局问题

CSS 给人的感觉就是 simple，但是前端开发者一定深有体会：simple 并不意味着 easy。我们这里不一一列举各种 CSS 「疑难杂症」，而是深入一个概念 —— BFC。BFC 是前端面试中的一个超级热点，今日头条某部门曾经就问过我：

请解释一下 BFC 是什么？

回答这个问题并不困难，但是我们可以继续追问：

BFC 会引起哪些布局现象？

这一小节，我们通过对 BFC 的分析，也顺带回顾一下那些 CSS 常考的小细节。

BFC 是什么

简单来说，BFC 就是：

BFC 是 Block Formatting Context 的简写，我们可以直接翻译成「块级格式化上下文」。它会创建一个特殊的区域，在这个区域中，只有 block box 参与布局。而 BFC 的一套特点和规则就规定了在这个特殊的区域中如何进行布局，如何进行定位，区域内元素的相互关系和相互作用。这个特殊的区域不受外界影响。

上面提到了 block box 的概念，block box 是指 display 属性为 block、list-item、table 的元素。

顺便插一个问题：那你还知道其他哪些 box 类型呢？

相应地，我们有 inline box，它是指 display 属性为 inline、inline-block、inline-table 的元素。CSS3 规范中又加入了 run in box，这里我们不再展开。

如何形成 BFC

那么什么样的情况会创建一个 BFC 呢？MDN 总结如下：

根元素或其他包含它的元素

浮动元素（元素的 float 不是 none）

绝对定位元素（元素具有 position 为 absolute 或 fixed）

内联块（元素具有 display: inline-block）

表格单元格（元素具有 display: table-cell，HTML 表格单元格默认属性）

表格标题（元素具有 display: table-caption，HTML 表格标题默认属性）

具有 overflow 且值不是 visible 的块元素

display: flow-root 的元素

column-span: all 的元素

BFC 决定了什么

我们上面谈到了 BFC 的一套规则，那么这些规则都有哪些呢?

内部的 box 将会独占宽度，且在垂直方向，一个接一个排列

box 垂直方向的间距由 margin 属性决定，但是同一个 BFC 的两个相邻 box 的 margin 会出现边距折叠现象

每个 box 水平方向上左边缘，与 BFC 左边缘相对齐，即使存在浮动也是如此

BFC 区域不会与浮动元素重叠，而是会依次排列

BFC 区域内是一个独立的渲染容器，容器内元素和 BFC 区域外元素不会形成任何干扰

浮动元素的高度也参与到 BFC 高度的计算当中

从这些规则中，我们至少能总结出一些关键要点，比如：

边距折叠

清除浮动

自适应多栏布局

这也是我选取 BFC 这个概念来剖析的原因，理解了 BFC，这些常见、常考知识点我们都可以融会贯通，具体来看下下面的场景。

BFC 实战应用

例题 1

给出如下代码：

我们得到布局如图：

请在不修改已有内容情况下，加入样式，实现自适应（`.left` 宽度固定，`.right` 占满剩下宽度）两栏布局。

我们来思考：根据 `BFC` 布局规则：「每个 `box` 水平方向上左边缘，与 `BFC` 左边缘相对齐。即使存在浮动也是如此」，因此 `.left` 和 `.right` 的左边相接触。出现如此布局结果并不意外。

同时，再想想 BFC 布局规则：「BFC 区域不会与浮动元素重叠，而是会依次排列」，因此我们可以使 .right 形成 BFC，来实现自适应两栏布局。如何形成 BFC 前面已经做过介绍了，于是添加：

```
.right {
    overflow: hidden;
}
```

就可以得到：

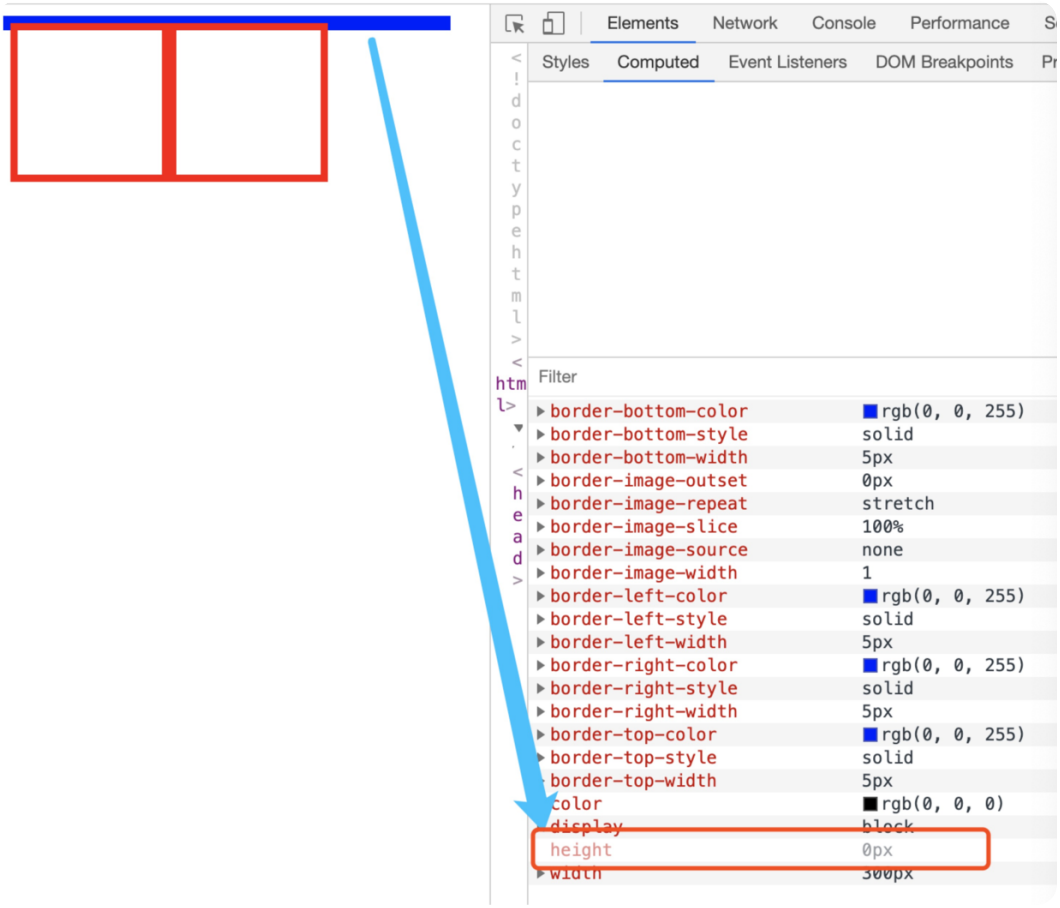
当然，这种布局可以用更先进的 flex 或者 grid 手段解决，但是对于 BFC 这些 CSS 基础知识，同样要做到了然于胸。

例题 2

看代码：

*首先来回答第一个问：**`.root` 的高度是多少?***

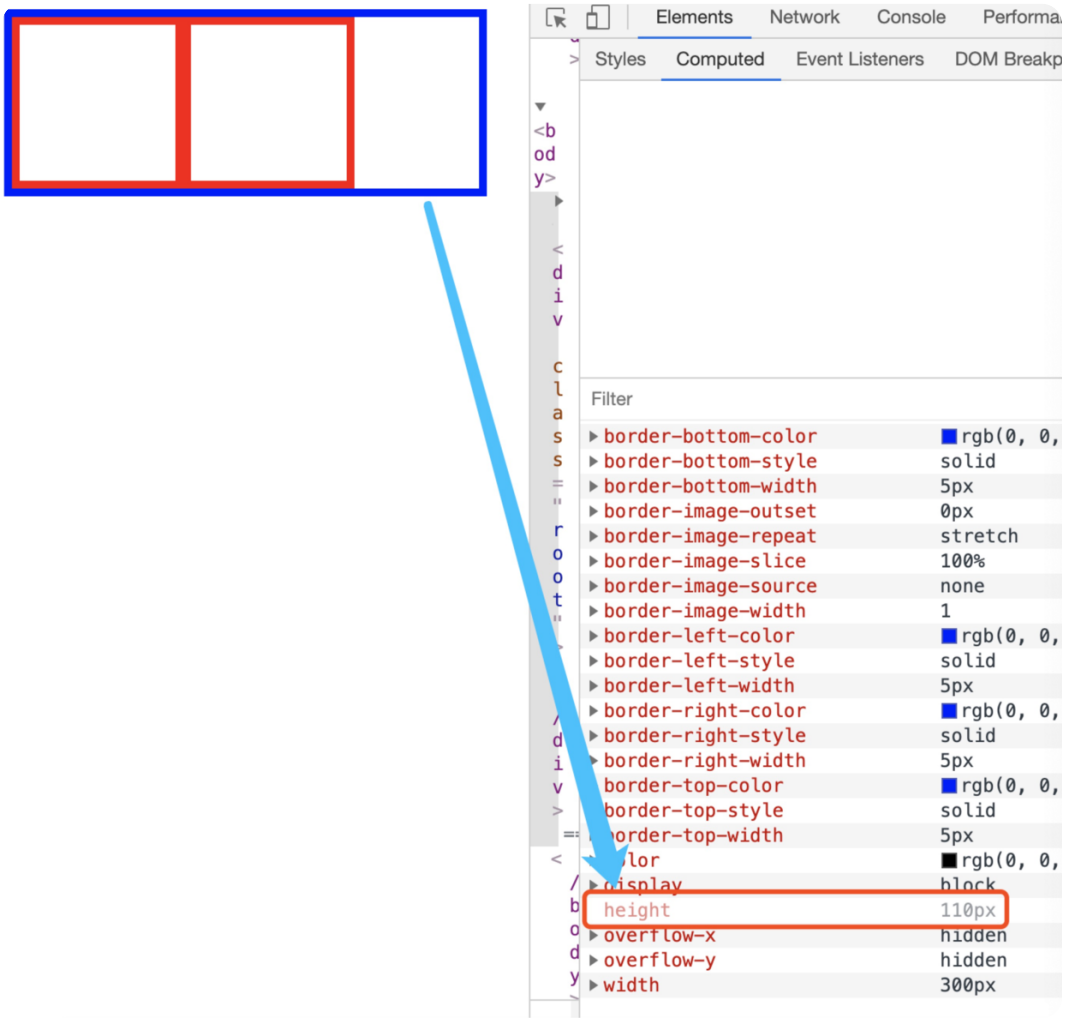
事实上，因为 `.child` 为浮动元素，因此造成了「高度塌陷」现象，`.root` 的高度为 `0`。



那么如何解决「高度塌陷」问题呢?

想想 BFC 规则：「浮动元素的高度也参与到 BFC 高度的计算当中」，因此使 .root 形成 BFC，就能解决问题：

```
.root {
    overflow: hidden;
}
```



我们看此时高度已经被你撑开了。

例题 3

代码：

```
paragraph 1
```

```
paragraph 2
```

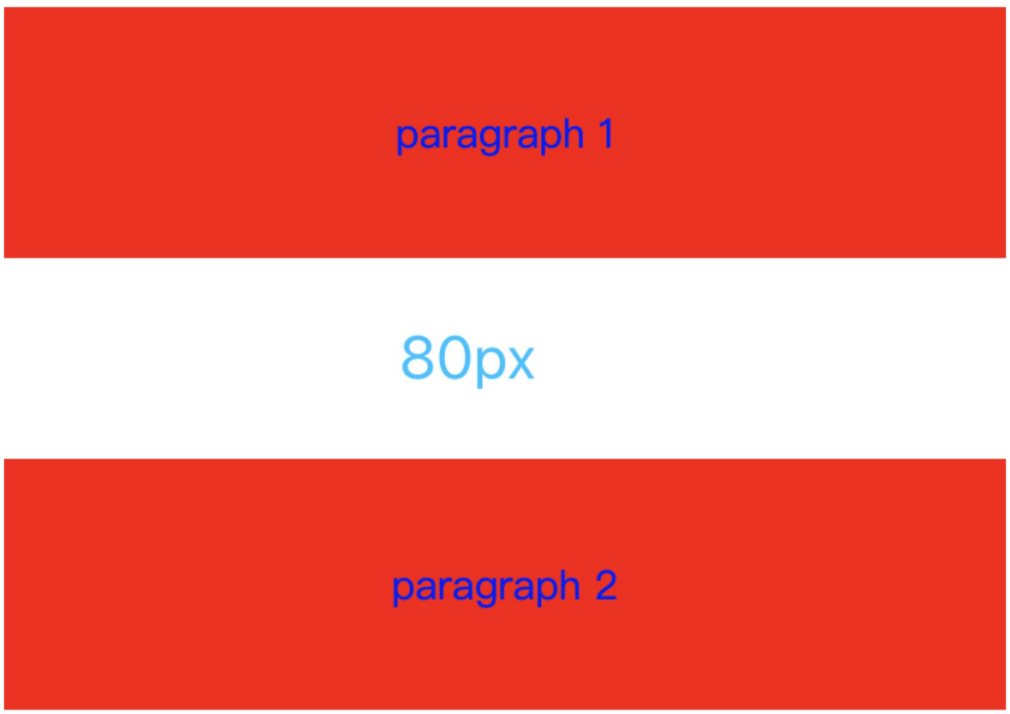
首先回答问题：**两段之间的垂直距离为多少？** 想想 BFC 规则：「box 垂直方向的间距由 margin 属性决定，但是**同一个**BFC 的两个相邻 box 的 margin 会出现边距折叠现象」。事实上，因为边距折叠现象，答案为 40px。

那么如何解决这个问题呢？

最简单地，我们可以在 p 标签再包裹一个元素，并触发该元素形成一个 BFC。那么这两个 p 标签，不再属于同一个 BFC，从而解决问题。

```
paragraph 1
```

```
paragraph 2
```



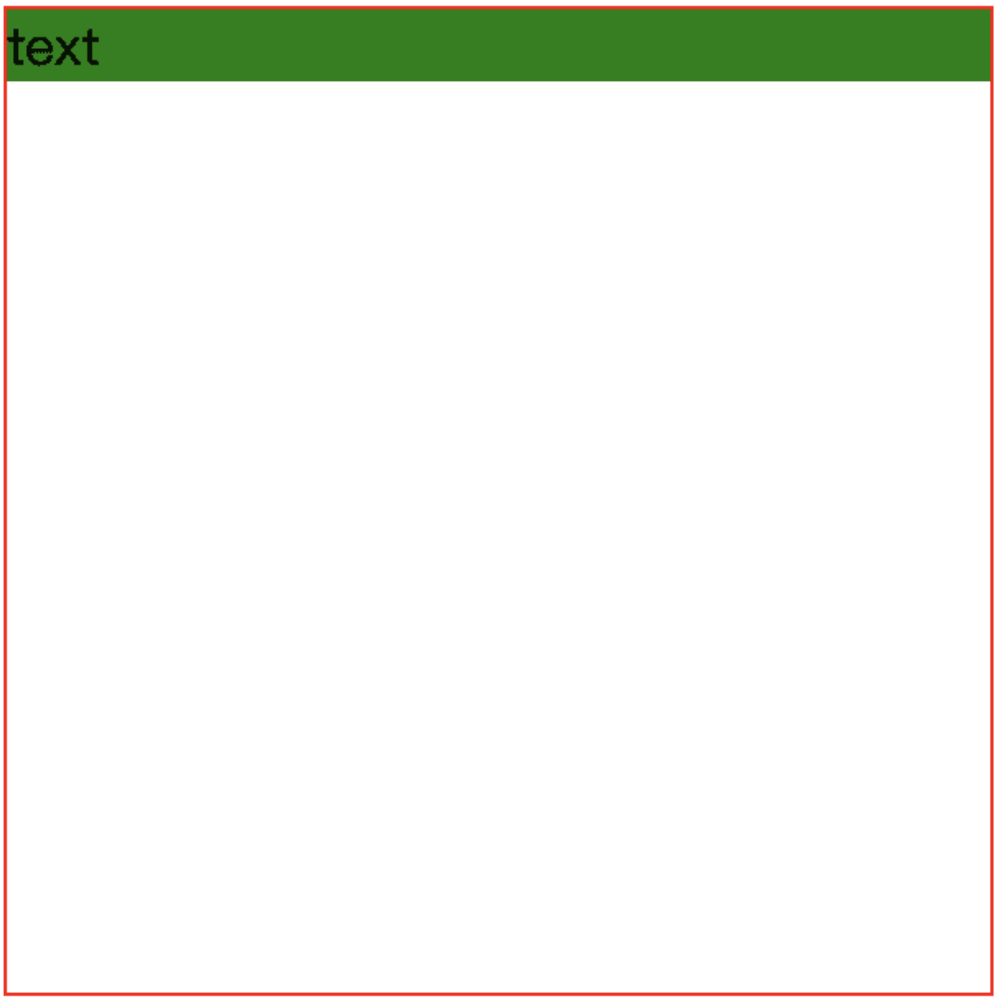
总结：*我们通过分析 BFC 是什么、如何形成、布局规则，融会贯通了 CSS 当中很多关键问题。也许不少读者能够解决「边距折叠」、「多栏自适应」、「高度塌陷」等问题，但是并不能说出解决问题的原理。通过这一环节的学习，我们对 CSS 加深了理解，我更希望地是能够启发大家思考：我们到底应该如何对待 CSS、如何学习 CSS。*

多种方式实现居中

「实现居中」也是一道必考题。参考代码：

```
text
```

如图：



如何让绿色的块水平垂直居中呢?

总结一下:

仅适用于居中元素定宽高

absolute + 负 margin

```
.wp {
    position: relative;
}
.box {
    position: absolute;;
    top: 50%;
    left: 50%;
    margin-left: -50px;
    margin-top: -50px;
}
```

绝对定位的百分比是相对于父元素的宽高，我们设置：

```
top: 50%;
left: 50%;
```

使得元素偏移后，在修正元素自身宽高的一半即可：

```
margin-left: -50px;
margin-top: -50px;
```

这其实是一个简单的数学几何运算。

absolute + margin auto

```
.wp {
    position: relative;
}
.box {
    position: absolute;;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    margin: auto;
}
```

这种方式将设置各个方向的距离都是 0，此时配合 margin 为 auto，就可以在各个方向上居中了。

absolute + calc

```
.root {
    position: relative;
}
.textBox {
    position: absolute;;
    top: calc(50% - 50px);
    left: calc(50% - 50px);
}
```

此种方法和第一种类似，不再展开。

居中元素不定宽高

对于剧中元素不定宽高的情况：

text

我们依然也有很多方法。

absolute + transform

不定宽高时，利用 CSS3 新增的 transform，transform 的 translate 属性也可以设置百分比，这个百分比是相对于自身的宽和高，因此可以将 translate 设置为 -50%：

```
.wp {
    position: relative;
```

```
}

.box {

    position: absolute;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%);

}
```

原理和第一种方法也类似。

lineheight

把 box 设置为行内元素，通过 text-align 也可以做到水平居中，同时通过 vertical-align 做到垂直方向上的居中，代码如下：

```
.wp {

    line-height: 300px;

    text-align: center;
```

```
    font-size: 0px;
}
.box {
    font-size: 16px;
    display: inline-block;
    vertical-align: middle;
    line-height: initial;
    text-align: left; /* 修正文字 */
}
```

这个方法充分利用了行内 / 块级元素的特点。

table

其实历史上 table 经常被用来做页面布局，这么做的缺点是会增加很多冗余代码，并且性能也不友好。不过处理居中问题，它可是能手：

```
test
```

```
.wp {
text-align: center;
}
.box {
display: inline-block;
}
```

css-table

如何使用 table 布局的特性效果，但是不采用 table 元素呢？答案是 css-table：

```
.wp {
    display: table-cell;
    text-align: center;
    vertical-align: middle;
}
.box {
    display: inline-block;
}
```

我们使用了 display: table-cell，同时和 table 布局相比，减少了很多冗余代码。

flex

flex 是非常现代的布局方案，只需几行代码就可以优雅地做到居中：

```
.wp {
    display: flex;
    justify-content: center;
    align-items: center;
}
```

grid

grid 布局非常超前，虽然兼容性不好，但是能力超强：

```
.wp {
    display: grid;
}
.box {
    align-self: center;
```

```
    justify-self: center;
}
```

我们总结一下：

PC 端有兼容性要求，宽高固定，推荐 absolute + 负 margin

PC 端有兼容要求，宽高不固定，推荐 css-table

PC 端无兼容性要求，推荐 flex

移动端推荐使用 flex

最后整理一个列表：

方法	居中元素定宽高固定	PC兼容性	移动端兼容性
absolute + 负margin	是	ie6+, chrome4+, firefox2+	安卓2.3+, iOS6+
absolute + margin auto	是	ie6+, chrome4+, firefox2+	安卓2.3+, iOS6+
absolute + calc	是	ie9+, chrome19+, firefox4+	安卓4.4+, iOS6+
absolute + transform	否	ie9+, chrome4+, firefox3.5+	安卓3+, iOS6+
writing-mode	否	ie6+, chrome4+, firefox3.5+	安卓2.3+, iOS5.1+
lineheight	否	ie6+, chrome4+, firefox2+	安卓2.3+, iOS6+
table	否	ie6+, chrome4+, firefox2+	安卓2.3+, iOS6+
css-table	否	ie8+, chrome4+, firefox2+	安卓2.3+, iOS6+
flex	否	ie10+, chrome4+, firefox2+	安卓2.3+, iOS6+
grid	否	ie10+, chrome57+, firefox52+	安卓6+, iOS10.3+

总结

HTML 和 CSS 面试中考察较少，但是如果答的不好，将是致命性的。同时工作中，如果这方面知识存在短板，往往会造成不必要的效率消耗。我们应该正视前端领域这两个离不开的「面子工程」，为了更好的面试结果，更为了自己的技能。

点击查看下一节

进击的 HTML 和 CSS