你知道的 CSS 垂直居中的方式有哪些

在 CSS 中对元素进行水平居中是非常简单的：如果它是一个行内元素， 就对它的父元素应用 `text-align:center;` 如果它是一个块级元素，就对它自身应用 `margin:auto`。然而如果要对一个元素进行垂直居中，可能光是想想就令人头皮发麻了。

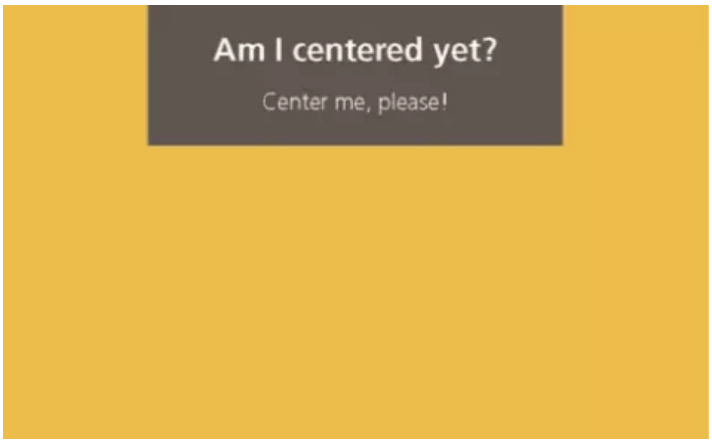
长久以来，为了解决这一绝世难题，前端开发者们殚精竭虑，琢磨出了各种解决方法，可惜大多数并不实用。在本篇攻略中，我们将探索现代 CSS 的强大威力，以全新的思路去攻克各种场景下的垂直居中难题。请注意，有几种技巧十分流行，但在这里并不会深入探讨，原因如下：

- 表格布局法(利用表格的显示模式)需要用到一些冗余的 HTML 元素，实现起来不够优雅简洁，因此这里不多介绍。
- 行内块法也不作讨论，因为在我看来这种方法 hack 的味道很浓。

如果你有兴趣，可以去看看 Chris Coyier 写的 Centering in the Unknown。这篇出色的文章详细讲述了这两种技巧。

我们将使用如下所示的结构代码，并直接插入 元素中（但实际上我们将要探索的这些技巧是与容器无关的）：

```
<main>
      <h1>Am I centered yet?</h1> <p>Center me, please!</p>
</main>
```



基于绝对定位

我们先来看一个早期的垂直居中方法，它要求元素具有固定的宽度和高度：

```
main {
      position: absolute;
      top: 50%;
      left: 50%;
      margin-top: -3em; /* 6/2 = 3 */ margin-left: -9em; /* 18/2 = 9 */ width: 18em;
      height: 6em;
}
```

这段代码在本质上做了这样几件事情：

1. 先把这个元素的左上角放置在视口（或最近的、具有定位属性的祖先元素）的正中心。
2. 然后再利用负外边距把它向左、向上移动（移动距离相当于它自身宽高的一半），从而把元素的正中心放置在视口的正中心。

借助强大的 `calc()` 函数，这段代码还可以省掉两行声明：

```
main {
      position: absolute;
      top: calc(50% - 3em);
      left: calc(50% - 9em);
      width: 18em;
      height: 6em;
}
```

显然，这个方法最大的局限在于它要求元素的宽高是固定的。

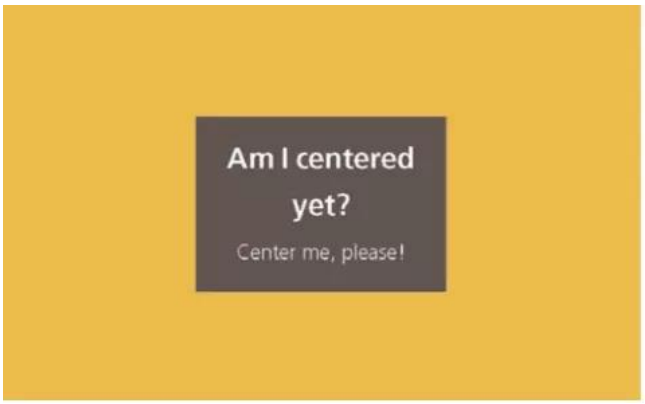
在通常情况下，对那些需要居中的元素来说，其尺寸往往是由其内容来决定的。如果 能找到一个属性的百分比值以元素自身的宽高作为解析基准，那我们的难题就迎刃而解了！遗憾的是，对于绝大多数 CSS 属性(包括 margin)来说， 百分比都是以其父元素的尺寸为基准进行解析的。

CSS 领域有一个很常见的现象，真正的解决方案往往来自于我们最意想不到的地方。

在这个例子中，答案来自于 CSS Transforms 。当我们在 `translate()` 变形函数中使用百分比值时，是以这个元素自身的宽度和高度为基准进行换算和移动的，而这正是我们所需要的。接下来，只要换用基于百分比的 CSS 变形来对元素进行偏移，就不需要在偏移量中把元素的尺寸写死。这样我们就可以彻底解除对固定尺寸的依赖：

```
main {
      position: absolute;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%);
}
```

可以看到，这个容器已经完美居中了，完全满足我们的期望。



当然，没有任何技巧是十全十美的，上面这个方法也有一些需要注意的地方：

- 我们有时不能选用绝对定位，因为它对整个布局的影响太过强烈。
- 如果需要居中的元素已经在高度上超过了视口，那它的顶部会被视口裁切掉。有一些办法可以绕过这个问题，但 hack 味道过浓。
- 在某些浏览器中，这个方法可能会导致元素的显示有一些模糊，因 为元素可能被放置在半个像素上。这个问题可以用 `transform-style:preserve-3d` 来修复，不过这个修复手段也可以认为是一个 hack，而且很难保证它在未来不会出问题。

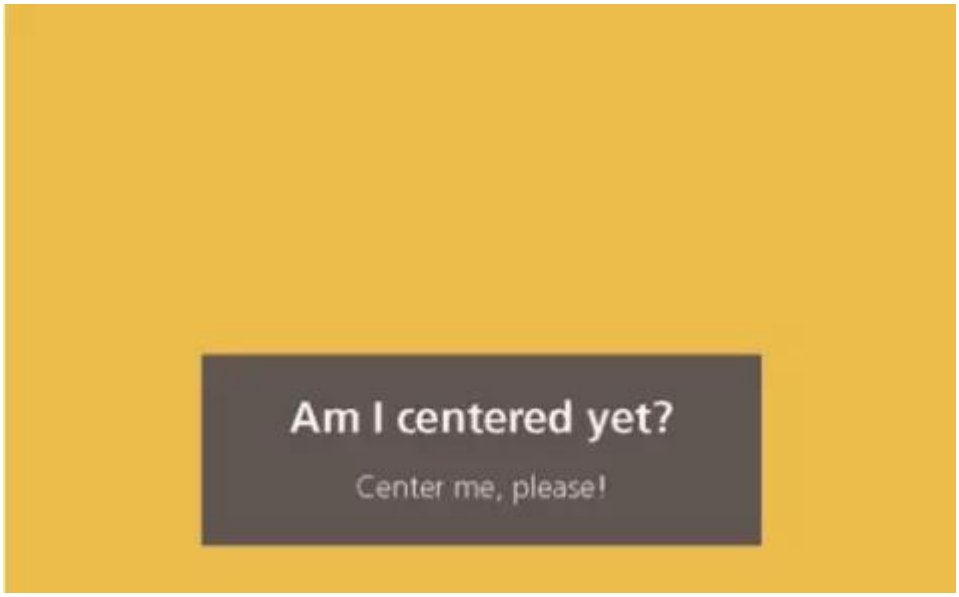
基于 viewport

假设我们不想使用绝对定位，仍然可以采用 `translate()` 技巧来把这 个元素以其自身宽高的一半为距离进行移动；但是在缺少 left 和 top 的情况下，如何把这个元素的左上角放置在容器的正中心呢？

我们的第一反应很可能是用 margin 属性的百分比值来实现，就像这样：

```
main {
        width: 18em;
        padding: 1em 1.5em;
        margin: 50% auto 0;
        transform: translateY(-50%);
}
```

不过，如图所示，这段代码会产生十分离谱的结果：



原因在于 **margin 的百分比值是以父元素的宽度作为解析基准的** 。没错，即使对于 `margin-top` 和 `margin-bottom` 来说也是这样！

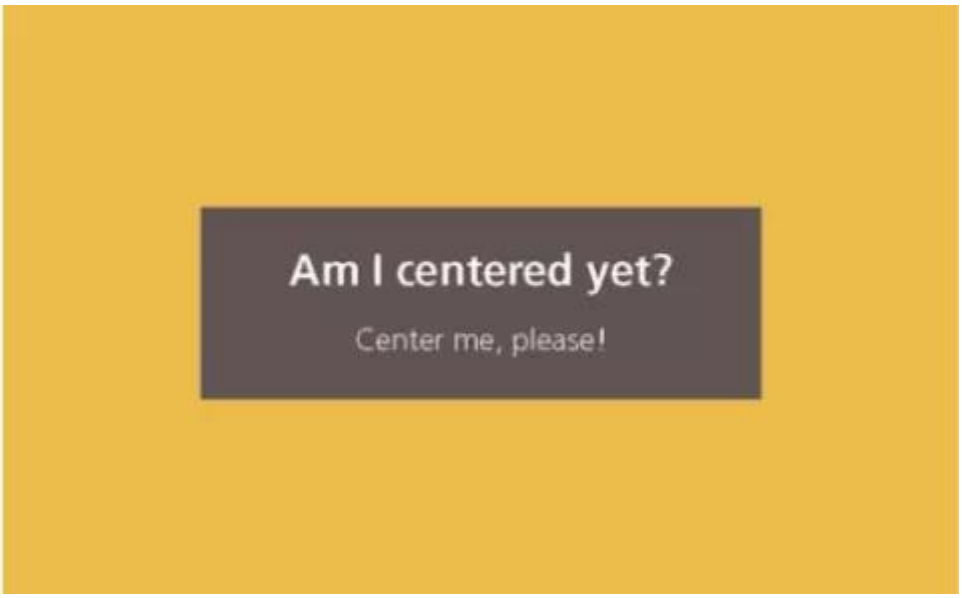
不过幸运的是，如果只是想把元素相对于视口进行居中，仍然是有希望的。【CSS Values and Units Module Level 3】(https://www.w3.org/TR/css-values-3/#relative-lengths) 定义了一套新的单位，称为 viewport 相关的长度单位。

- vw ： 1% of viewport’s width
- vh ： 1% of viewport’s height
- vmin ： 1% of viewport’s smaller dimension
- vmax ： 1% of viewport’s larger dimension

在我们的这个例子中，适用于外边距的是 vh 单位：

```
main {
        width: 18em;
        padding: 1em 1.5em;
        margin: 50vh auto 0;
        transform: translateY(-50%);
}
```

可以看到，其效果堪称完美。当然，这个技巧的实用性是相当有限的，因为它只适用于在视口中居中的场景。



基于 Flexbox

这是毋庸置疑的最佳解决方案，因为 Flexbox 是专门针对这类需求所设计的。我们之所以要讨论其他方案，仅仅是因为那些方案在浏览器的支持程度上稍微好一些而已。但其实目前现代浏览器对 Flexbox 的支持度已经相当不错了。

我们只需写两行声明即可：

- 先给这个待居中元素的父元素设置 `display:flex`（在这个例子中是 `<body>` 元素）。

- 再给这个元素自身设置我们再熟悉不过的 `margin:auto`（在这个例子中是 `<main>` 元素）。

```
body {
      display: flex;
      min-height: 100vh;
      margin: 0;
}
main {
      margin: auto;
}
```

请注意，**当我们使用 Flexbox 时，margin: auto 不仅在水平方向上将元素居中，垂直方向上也是如此**。还有一点，我们甚至不需要指定任何宽度(当然，如果需要的话，也是可以指定的)：这个居中元素分配到的宽度等于 `max-content`。

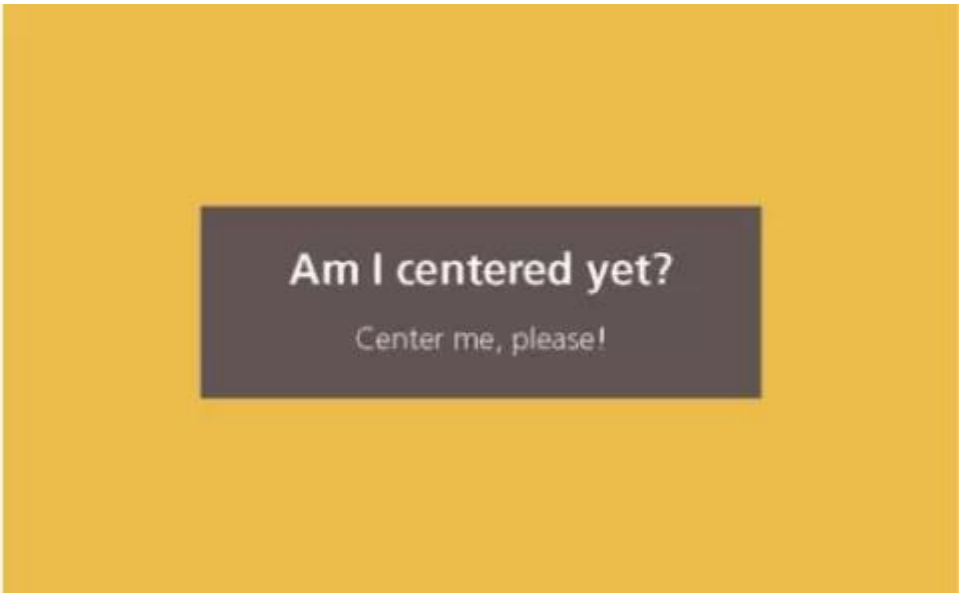
如果浏览器不支持 Flexbox，页面渲染结果看起来就跟我们的起点图是一样的了(如果设置了宽度的话)。虽然没有垂直居中效果，但也是完全可以接受的。

Flexbox 的另一个好处在于，它还可以将匿名容器(即没有被标签包裹的文本节点)垂直居中。举个例子，假设我们的结构代码是：

```
<main>Center me, please!</main>
```

我们先给这个 main 元素指定一个固定的尺寸，然后借助 Flexbox 规范所引入的 `align-items` 和 `justify-content` 属性，我们可以让它内部的文本也实现居中：

```
main {
      display: flex;
      align-items: center; justify-content: center; width: 18em;
      height: 10em;
}
```


Am I centered yet?
Center me, please!