45 响应式布局原理与实践（上）

更新时间：2020-06-15 09:51:12

> 梦想只要能持久，就能成为现实。我们不就是生活在梦想中的吗？——丁尼生

响应式布局是 CSS 布局中一个特别的分支。基于响应式布局，可以衍生出一个非常完整的移动端适配的知识链路。

因此，响应式布局在面试中，绝不是一个两三句话就能讲完的考点。接下来的两节，笔者会围绕响应式布局，为大家梳理出一个完整的知识脉络，帮助大家做到思路上有迹可循、表达上言之有物。

在本节，我们重点掌握响应式布局相关的两个重点概念：viewport 和 rem、em。这两块知识点非常有意思：大部分的候选人在回答响应式布局方案时能够行云流水，但涉及 viewport 和 rem、em 相关的解释和辨析时却只能支支吾吾、闪烁其词。俗话说基础不牢，地动山摇。在学习布局方案之前，我们首先要对核心概念形成自己的理解。

理解 viewport

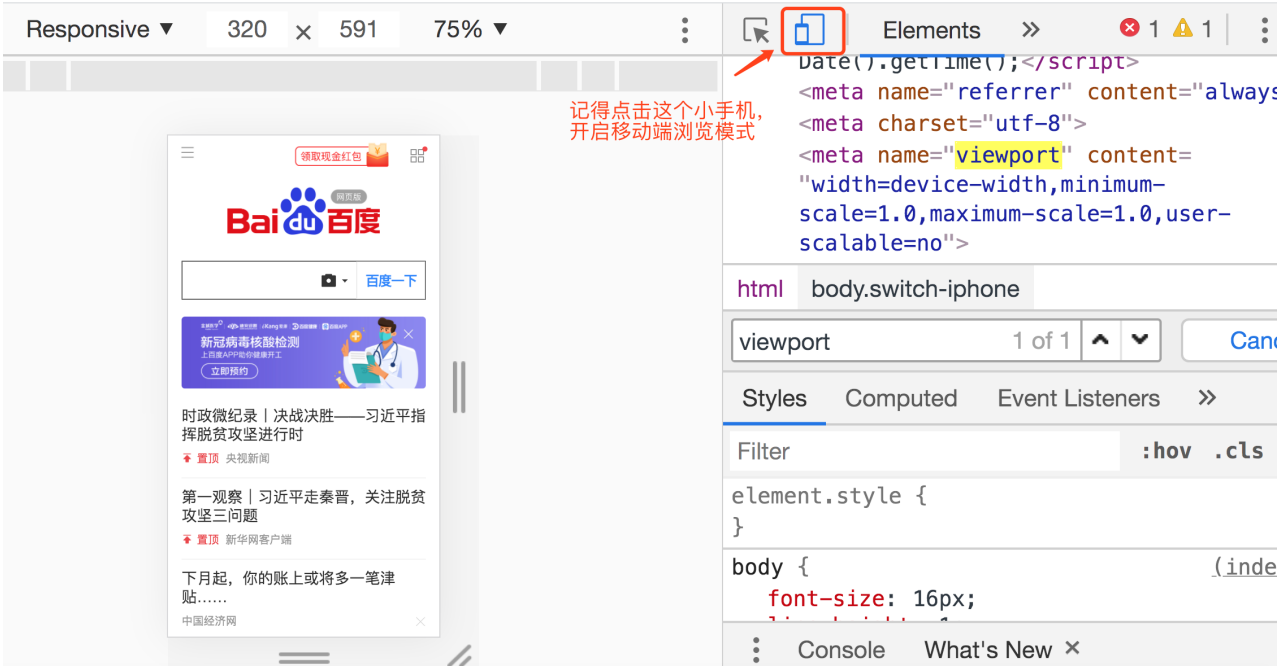
viewport 也叫“视口”。在第一节中，大家已经初步见识到了 viewport 的编码形态，知道可以通过 meta 标签来控制它。但其实，这只是视口这个概念的一种形态。在移动端，大家需要了解三种视口：

布局视口（layout viewport）与视觉视口（visual viewport）

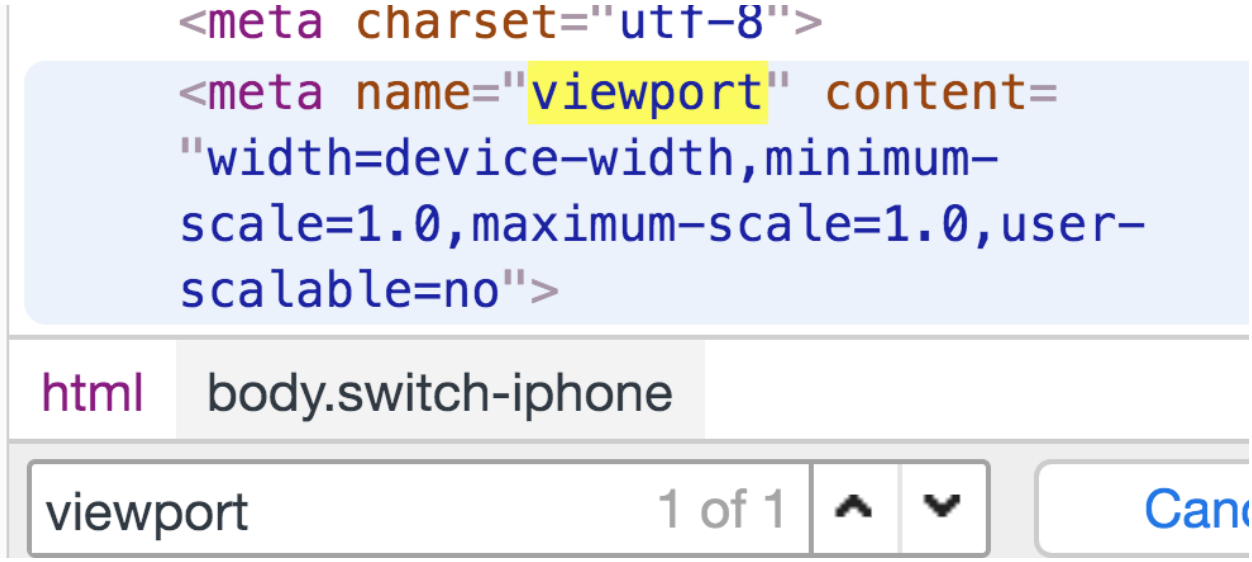
既然说到了 meta 标签里的 viewport，那我们就先拿它来开刀。在移动端开发中，我们经常可以见到这样的 meta 标签写法：

```
<meta name="viewport" content="width=device-width">
```

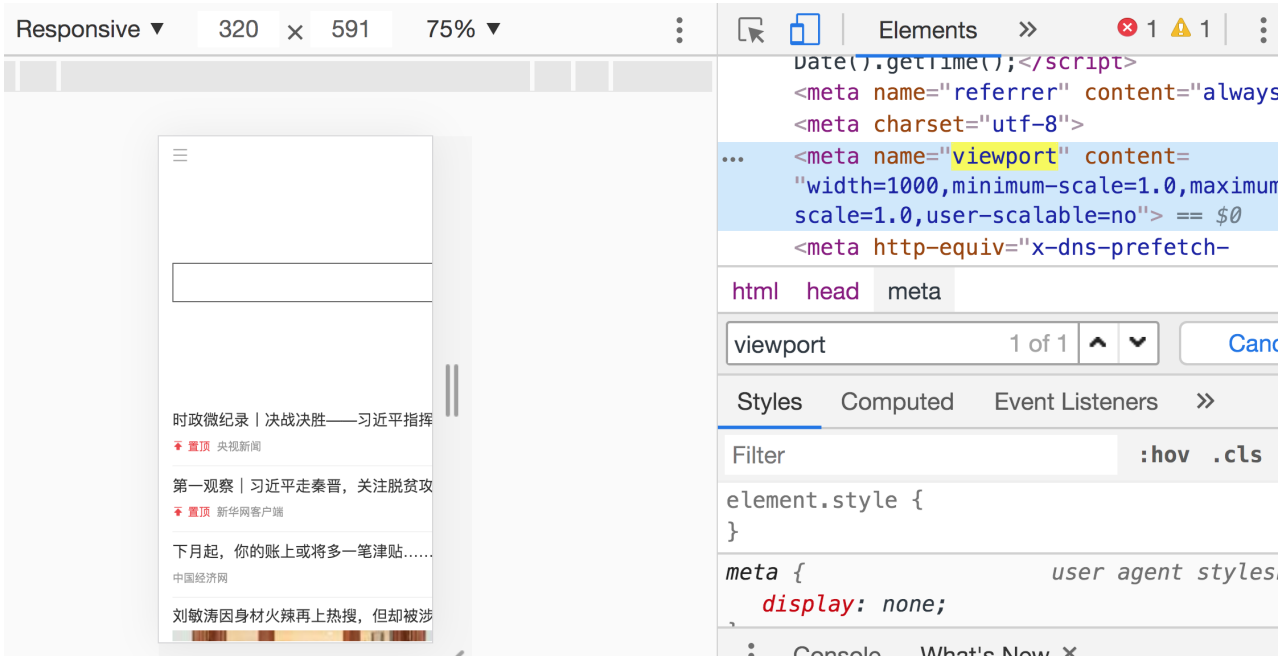
这里设置的 width，就是布局视口的宽度。那么什么是布局视口呢？大家现在可以打开一个新的 chrome 标签页，访问 https://www.baidu.com/（百度首页），点击右键，选择“检查”，选中 mobile 模式，然后刷新页面，你会得到一个类似这样的元素审查界面：



对 html 代码进行检索，我们会发现百度的移动端主页也在 meta 标签里对 viewport 进行了这个常见设置：



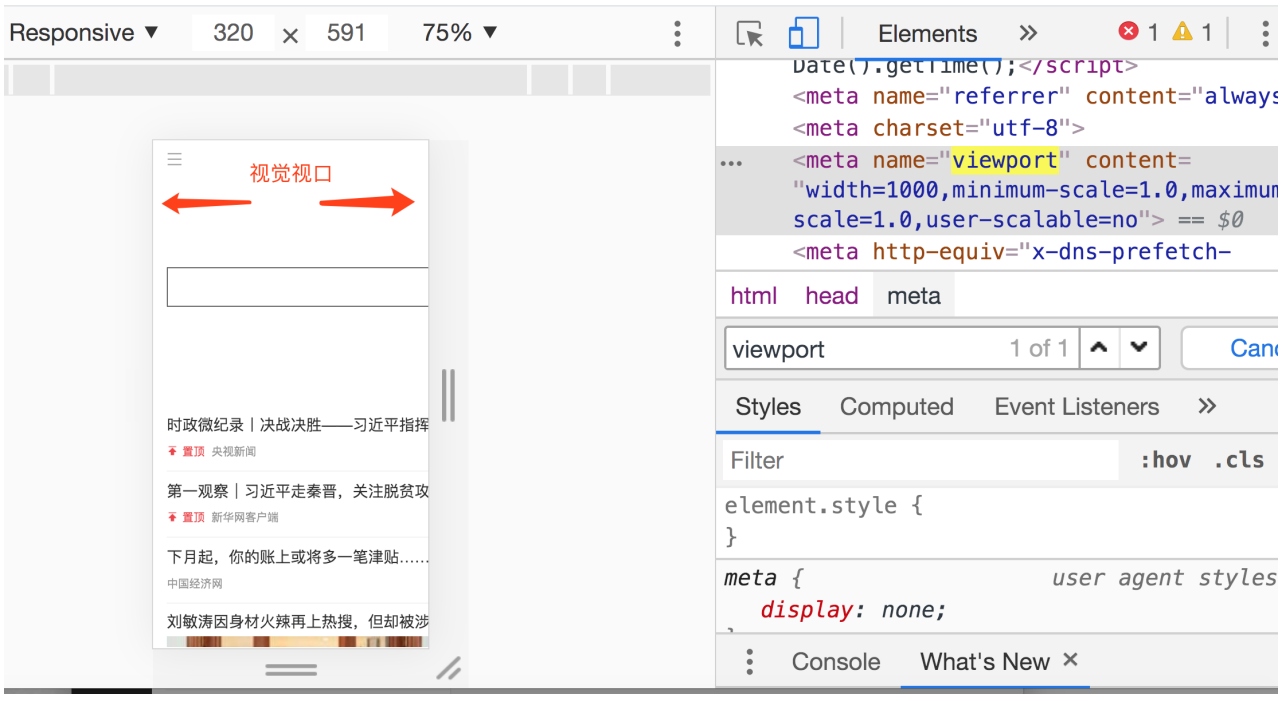
为什么一定要设置 width=device-width ？不设置会有什么后果？这里我随便把 width 改写成1000，发现页面变成了这样：



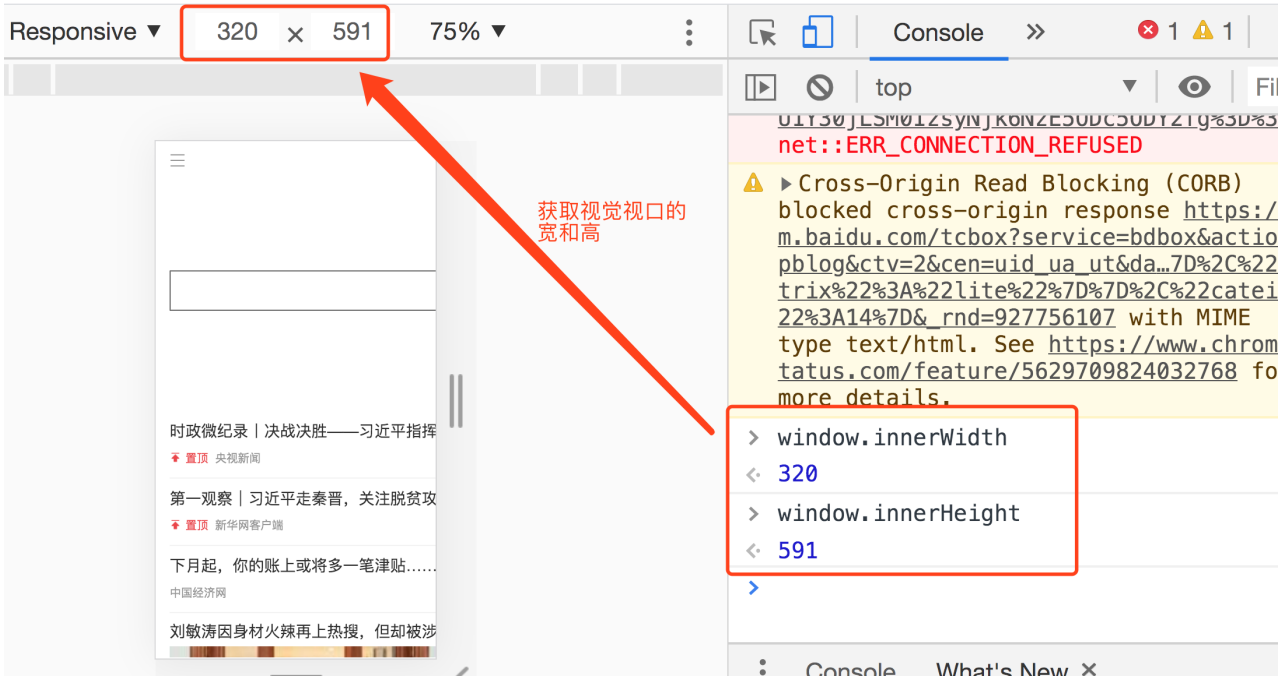
我们注意到，页面的可视区域变小了，这是为什么呢？

这是由于视觉视口和布局视口不相等导致的。

所谓视觉视口，它指的是你的设备实际的可见区域，也就是浏览器的宽高。在PC端，浏览器的宽高我们可以任意缩放；但在移动端，浏览器的宽高一般是不支持改变的，其大小由设备屏幕的大小决定。在当前的示例中，视觉视口就是下图红色箭头所标识的范围：

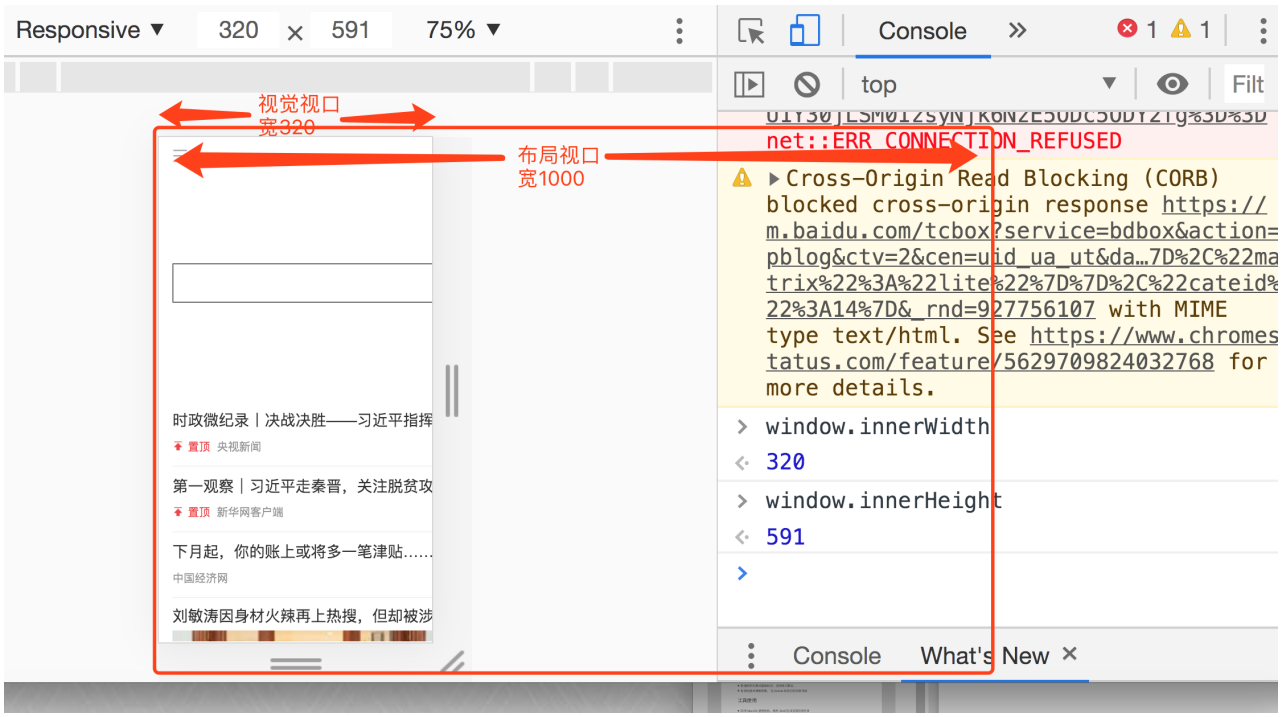


通过访问 `window.innerWidth` 和 `window.innerHeight` 两个属性，我们可以获取到视觉视口的宽高：

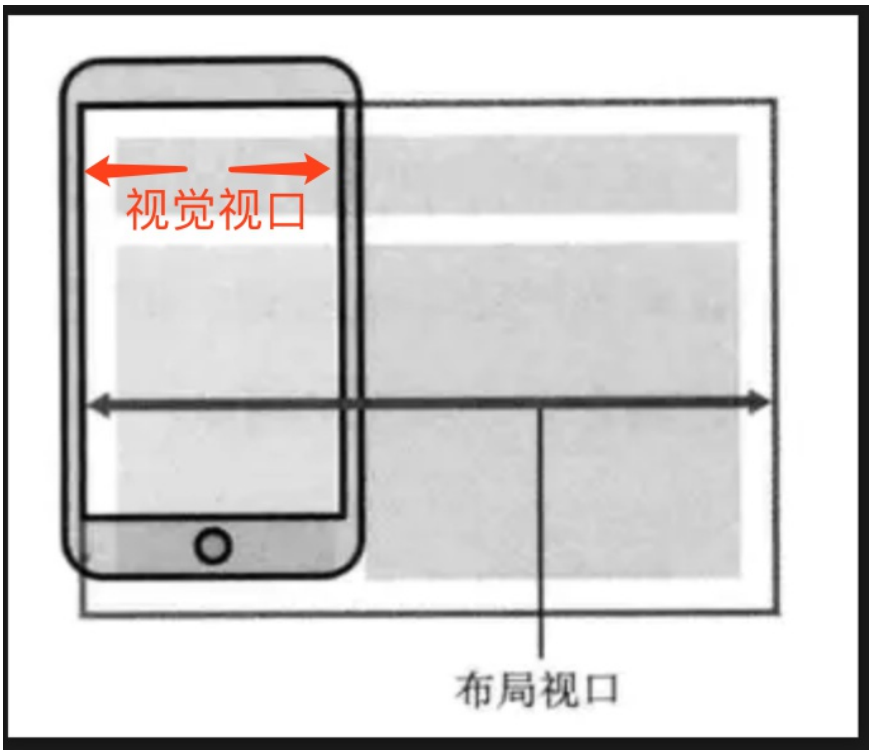


那么布局视口又是什么呢？

布局视口指的是页面实际布局所占用的区域。关于布局视口，网络上流传的定义有很多，大部分并不利于初学者的理解。这里我建议大家把布局视口理解为“画纸”，画纸的大小约束了你能够在多大的范围内作画。在本文的例子里，我们将布局视口的宽度从 device-width 改为 1000，这个1000其实是大于 device-width 的（为什么是大于，在理想视口部分我们会解释），也就是说“画纸”的区域超出了可见区域（视觉视口）的范围，因此我们最终只能从视觉视口中窥到其中的一角。此处视觉视口和布局视口的关系可以表达如下：



视觉视口和布局视口的关系可以进一步抽象如下：



我们可以通过 `document.documentElement.clientWidth` 来获取布局视口的宽度。

理想视口（**ideal viewport**）

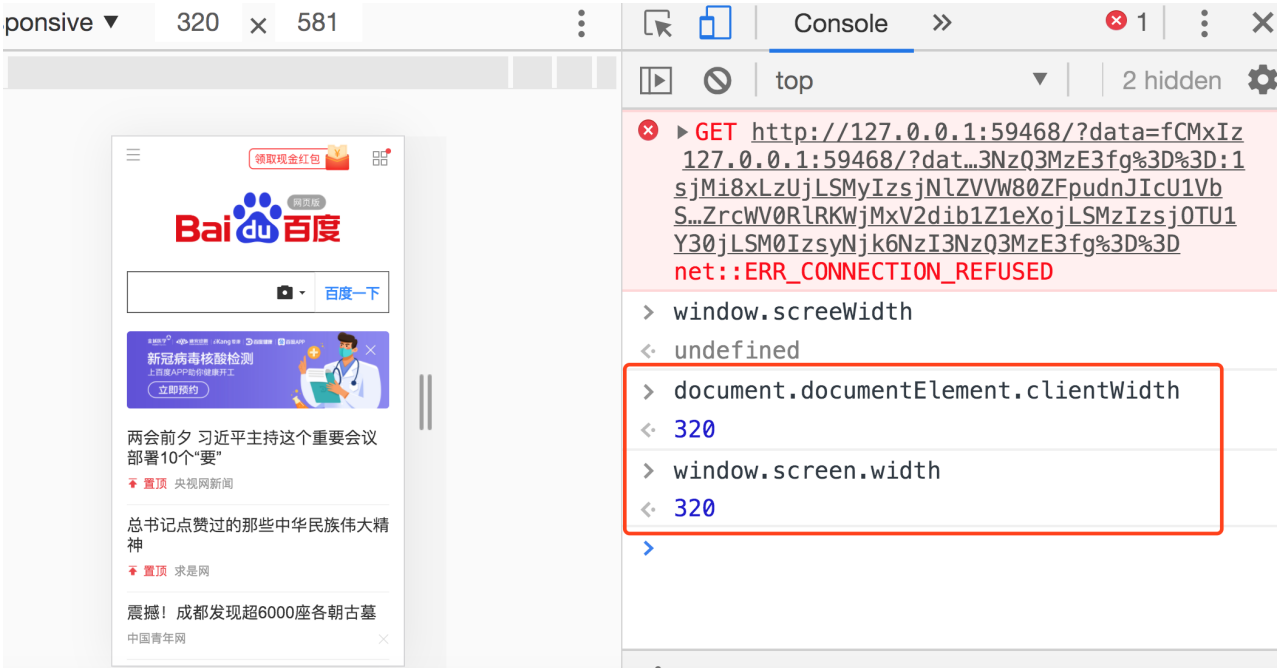
通过上面的解读我们可以发现，视觉视口和布局视口的大小并不总是一样的。当两者不等时，就会出现不符合预期的展示效果。实际上，很多时候，布局视口本身的宽度都是无法和视觉视口完全匹配的。为了解决这个问题，人们提出了“理想视口”的概念，它指的是布局视口最理想的尺寸。

“理想的尺寸”指的是整个页面刚好全部覆盖手机屏幕的尺寸。这个尺寸不需要我们手动计算，厂商根据手机屏幕尺寸大小，会提供一个最符合这个屏幕尺寸页面设计方案，我们通过这样一行代码就可以应用这个方案：

```
<meta name="viewport" content="width=device-width">
```

此处 `width` 属性对应的就是布局视口的值。设置 `width = device-width` 的目的，正是为了使布局视口的宽度刚好匹配上视觉视口的宽度。

此时我们再来尝试获取页面布局视口的宽度：



此时我们发现布局视口和视觉视口已经可以完美匹配上了。此时页面也恰恰好能够完全覆盖设备屏幕。

理解 rem 和 em

明确概念

什么是 rem

rem 指的是**相对于HTML根元素的字体大小**（font-size）来计算的长度单位。比如说我给 html 元素设置一个 font-size 是 100：

```
html {
  font-size: 100px;
}
```

那么就有如下的换算关系：

```
1rem = 100px
```

此时假如我们给一个 div 设置这样的样式:

```
div {
  width: 1rem;
  height: 2rem;
}
```

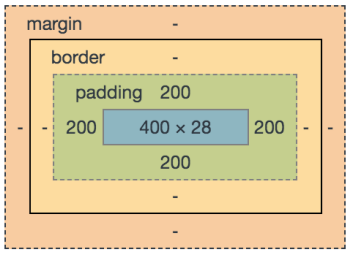
那么这个 div 的宽就是 100px，高就是 200px。

什么是 em

em 也是一个相对长度单位，它相对的是使用他们的元素的字体大小。比如我们给一个 div 设置字体大小为 20px：

```
div {
  font-size: 20px;
  padding: 10em;
  width: 20em;
}
```

那么对这个元素来说，就有 1em = 20px 的换算关系。按照这个换算关系，其 padding 值为 200px，width 值为 400px。实际展示出来也确实如此：



Filter Show al

display	block
font-size	20px
height	28px
padding-bottom	200px
padding-left	200px
padding-right	200px
padding-top	200px
width	400px

易错点辨析

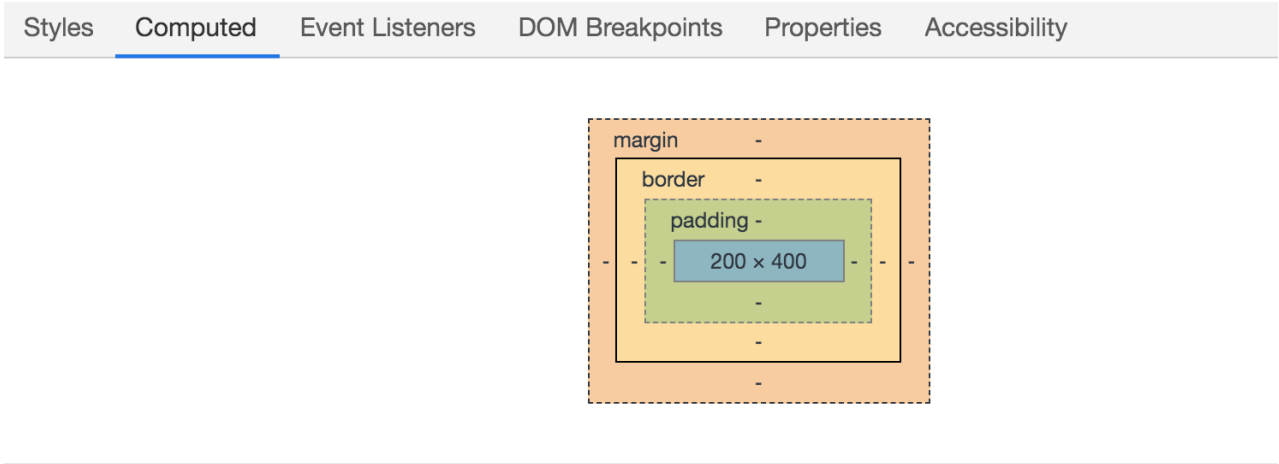
em 与继承相结合

这里有一个非常普遍的误区：很多候选人认为 em 单位取的是当前元素对应父元素的字体大小，这是不对的。的确，在一些情况下，em 确实会刚好取到父元素的字体大小。但这并非 em 的本色，而是由继承导致的。

举个例子：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>em继承示例</title>
  <style>
    #dad {
      font-size: 20px;
    }
    #son {
      width: 10em;
      height: 20em;
    }
  </style>
</head>
<body>
  <div id="dad">
    我是爸爸
    <div id="son">
      我是儿子
    </div>
  </div>
</body>
</html>
```

在这个示例中，我们没有指明子元素 son 的 font-size，根据 font-size 的继承特性，son 会继承 dad 的 font-size，也就是 20px。因此，son 这个元素的宽高分别为 200px 和 400px：

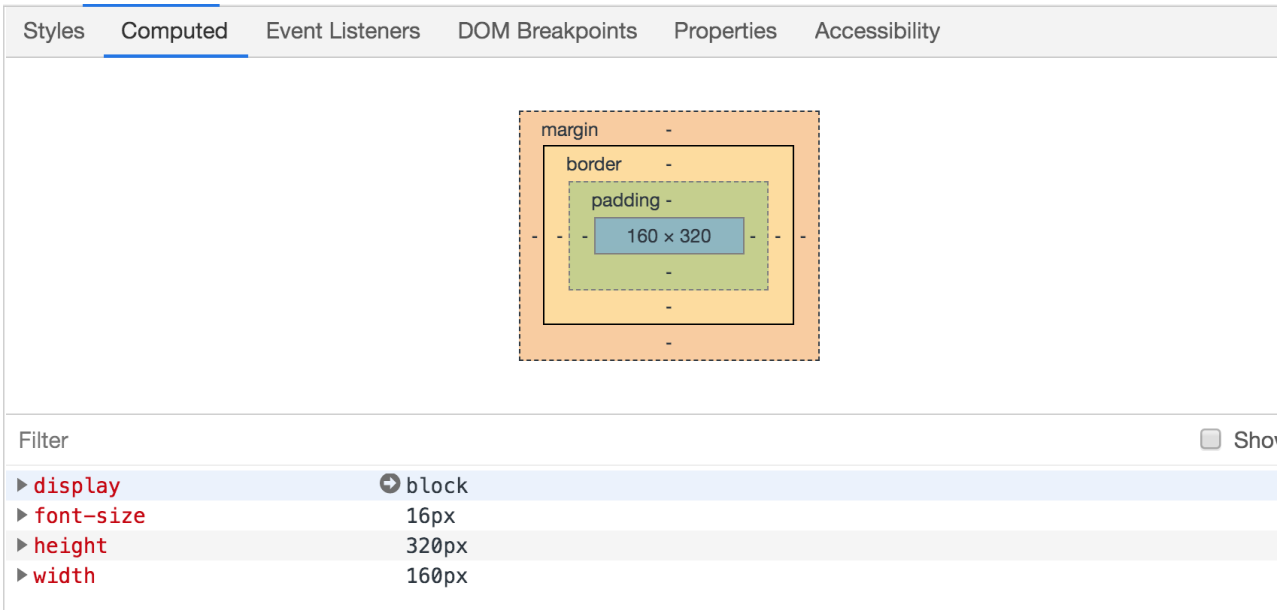


注意，此时 em 取的仍然不是父元素的 font-size，而是当前元素的 font-size。只是因为继承的发生，导致当前元素的 font-size 和 父元素的 font-size 值一样而已。

此时如果我给 son 一个自有的 font-size 属性，使继承不再发生，em 就会直接取我指定的这个 font-size 值了：

```
#son {
  font-size: 16px;
  width: 10em;
  height: 20em;
}
```

此时 son 元素中的 em 就是 16px：



根据以上两个例子我们可以看出，不管元素本身有没有显式地设置 **font-size**，**em** 取的都是当前元素的 **font-size**。

}

44 重点布局方案（下）
46 响应式布局原理与实践（下）