offsetLeft深入研究

基础示例代码-祖先元素无定位元素

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>offsetLeft研究</title>
</head>
<body style="border: 20px solid #000;">
    <div id="box1" style="border: 20px solid #666; width:300px; height:300px; padding-left:20px; margin-left:20px;">
        <div id="box2" style="border: 20px solid #999; width:200px; height: 200px; padding-left:20px; margin-left:20px;">
            <div id="box3" style="border: 20px solid #bbb;width:100px; height: 100px; padding-left:20px; margin-left: 20px; overflow:scroll; white-space: nowrap;">
                <div id="box4">这里是内容区域,父级为box3,父级超出部分滚动</div>
            </div>
        </div>
    </div>
</body>
<script>
    function clickFun(e){
        console.log('鼠标点击点的ClientX:', e.clientX);
        console.log('鼠标点击点的layerX:', e.layerX);
        console.log('鼠标点击点的pageX:', e.pageX);
        console.log('鼠标点击点的offsetX:', e.offsetX);
        console.log('鼠标点击点的screenX:', e.screenX);
        console.log('鼠标点击点的x:', e.x);
        console.log('box1的offsetLeft', $('box1').offsetLeft);
        console.log('box2的offsetLeft', $('box2').offsetLeft);
        console.log('box3的offsetLeft', $('box3').offsetLeft);
        console.log('box4的offsetLeft', $('box4').offsetLeft);
    }

    function $(id){
        return document.getElementById(id);
    }

    function addListener(type, fun) {
        if (typeof window.addEventListener != "undefined") {
                window.addEventListener(type, fun, false);
        } else {
            window.attachEvent('on' + type, fun)
        }
    }
    window.onload = function(){
        addListener('click', clickFun);
    };


```

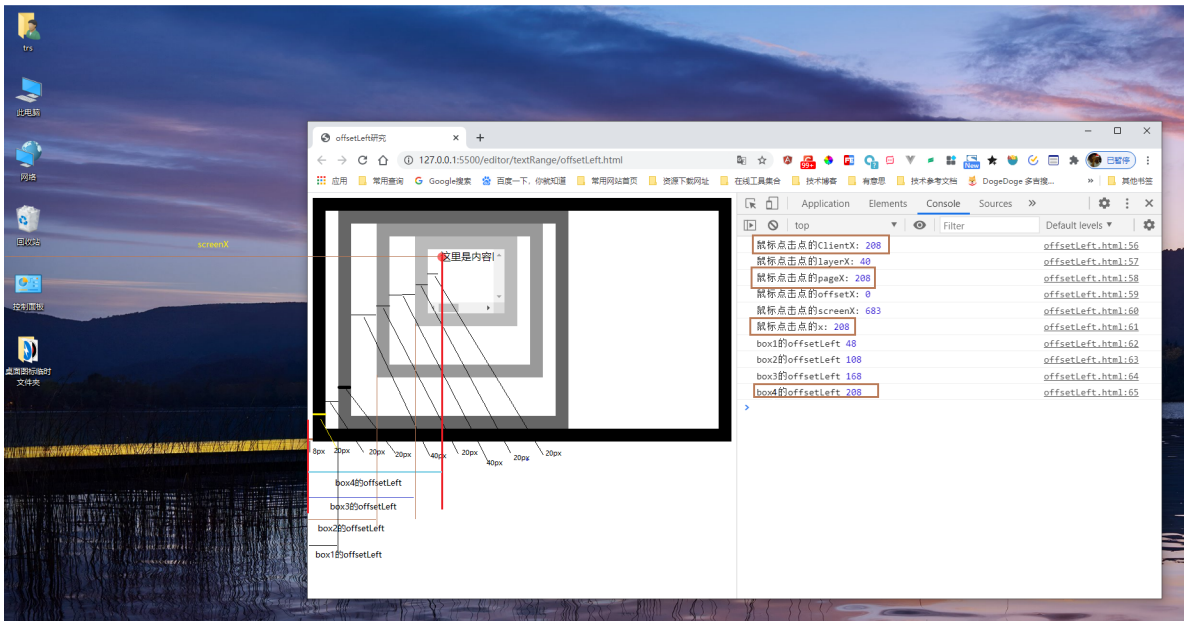
```
</script>
</html>
```

我们设计了实验代码结构，由 `body` 中的四层嵌套盒子组成，除最里层的 `box4` 外每个盒子都有20像素宽度的边框，20像素的左内边距，和20像素的左外边距。

当鼠标点击页面时，控制台输出当前点击位置的一些位置数据（只研究横向的，所以只打印横坐标值），同时打印输入四层盒子的 `offsetLeft` 的值。

同时， `box3` 设计了内容宽度超出容器后显示滚动条，以便验证滚动条对 `offsetLeft` 的影响。

先看默认状态下点击 `box4` 的左侧边框位置后输出的结果：



将截图放大观察，得出我们的

结论1：在祖先元素中没有定位元素时，元素的 `offset` 值为其左边框的最左侧位置到 `html` 页面左边框的距离值

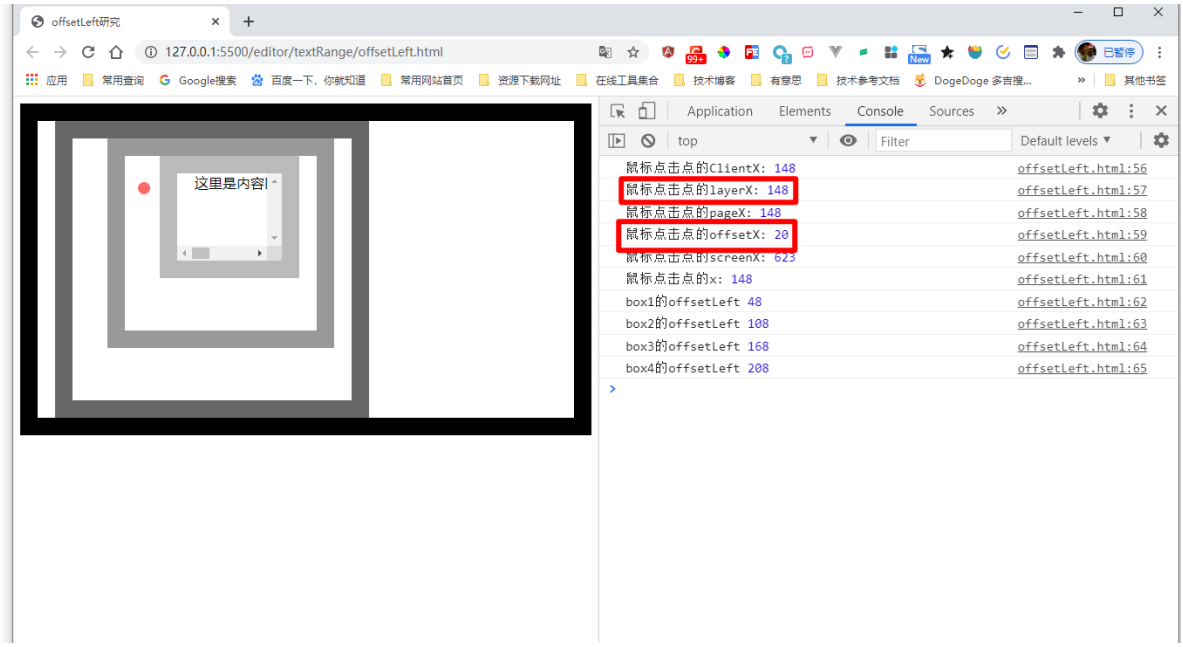
一点引申

根据输出结果，我们还可以猜想以下两个值的意义：

`offsetX`： 根据值的大小猜测为当前位置距离当前点击对象（ `e.target` ，这里为 `box4` ）左边框的值，至于是左边框的左侧还是右侧，需要进一步验证；

`layerX`：当前点击位置距离当前点击元素（ `e.target` ，这里为 `box4` ）父级元素左边框左侧的距离值

我们验证一下，这次点击 `box2` 左边框和 `box3` 左边框中间的位置:



可以看到，此时的 `offset` 值为20，正好是当前点击对象（ `box2` ）的左边框右侧位置到当前点击位置的距离。

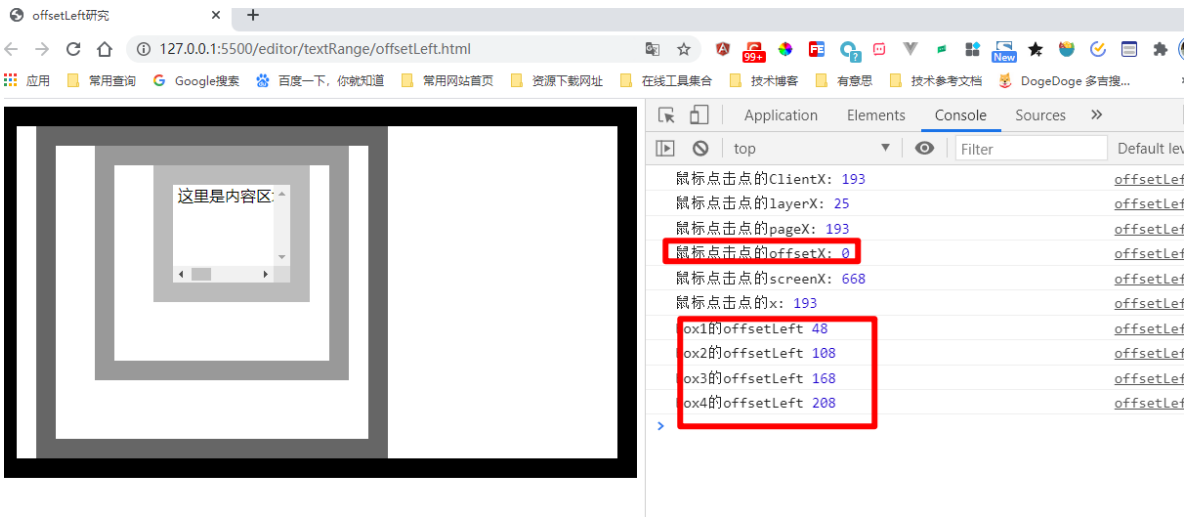
而此时的 `layerX` 值为148，并不是当前点击位置距离当前点击对象父级元素 `box1` 的左边框左侧的距离值，与我们的猜想不符，我们看看MDN的说明：

> The `**UIEvent.layerX**` read-only property returns the horizontal coordinate of the event relative to the current layer.
>
> `UIEvent.layerX` 只读属性返回事件相对于当前层的水平坐标。
>
> This property takes scrolling of the page into account and returns a value relative to the whole of the document unless the event occurs inside a positioned element, where the returned value is relative to the top left of the positioned element.
>
> 此属性将页面滚动考虑在内，并返回一个相对于整个文档的值，除非事件发生在定位元素内，其中返回值相对于定位元素的左上角

按照该说明，这个属性返回事件相对于当前层的坐标，那么很明显，当前层并不是事件目标元素的父元素，对于层的概念，大概和盒子的渲染模型有关，没有深入研究过，鉴于这个属性并不是W3C标准属性，各个浏览器实现也没有统一标准，不再深究。

滚动的影响

为了验证滚动是否对 `offsetLeft` 的值，我们拖动 `box3` 的滚动条，然后同样点击 `box4` 的左边框位置，看下结果：



可以看到，滚动条滚动后， `box4` 在页面上的绝对位置时改变了的，但是它（以及祖先元素）的 `offsetLeft` 属性并没有变化，得到

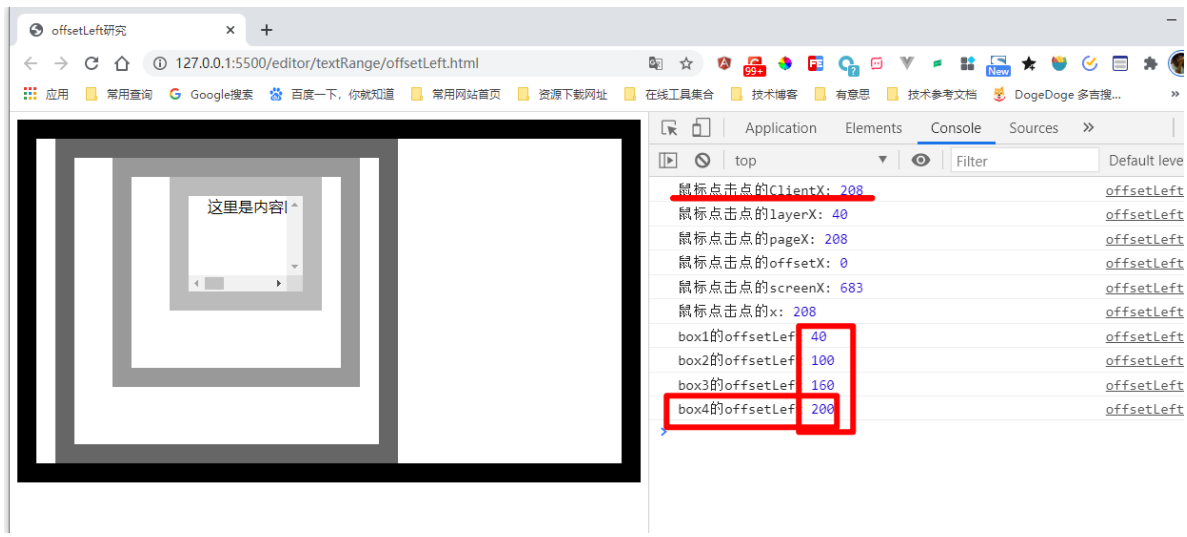
结论2： 滚动区域内的内容滚动后，其 `offsetLeft` 不受影响。

body元素定位

因为 `offsetLeft` 是相对于其 `offsetParent` 属性指向的元素（离它最新的定位祖先元素）的偏移量，所以必须研究当它的 `offsetParent` 发生改变时它的值的变化。首先，为 `body` 添加相对定位。

```
1 <body style="border: 20px solid #000; position: relative;">
```

使用变量控制实验法，仍旧点击 `box4` 的左边框，看看控制台输出：



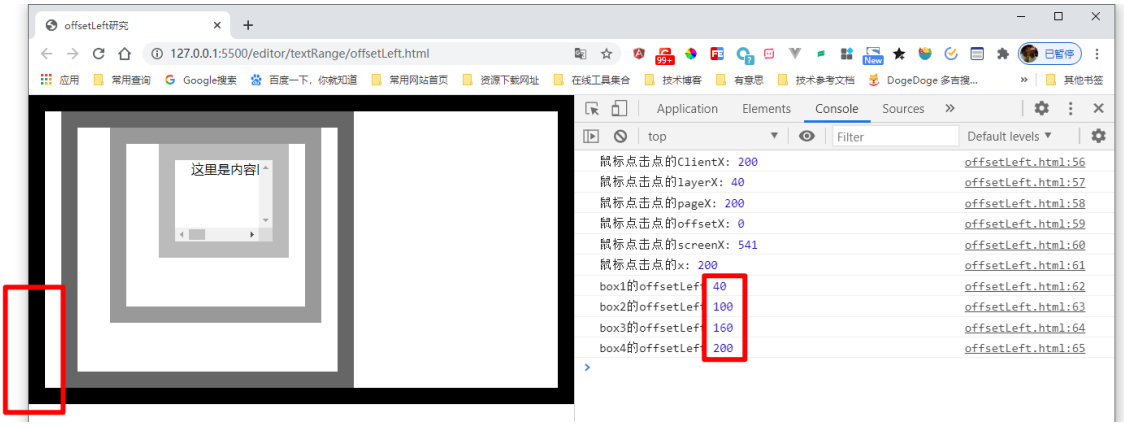
这次比我们第一次点击，位置时相同的（`clientX`都是208），但是所有元素的`offsetLeft`的值都少了8px，而这正好是我们第一张图中标识的`body`元素左边框左侧到`html`边框的距离。

这种变化是因为一开始`box4`的祖先元素中没有定位元素，所以其默认定位元素时`body`，但此时的`offsetLeft`包含了`body`的`margin-left`值（在`chrome`中默认为**8px**），而此时`body`被明确定义为了定位元素，此时的`offsetLeft`则不再包含`body`的`margin-left`值。

我们在代码中把`body`的`margin`值去掉，来验证下以上说法：

```
<body style="border: 20px solid #000; position: relative;margin:0;">
```

然后再看：



这里没有了`margin`值，所以我们修正一下

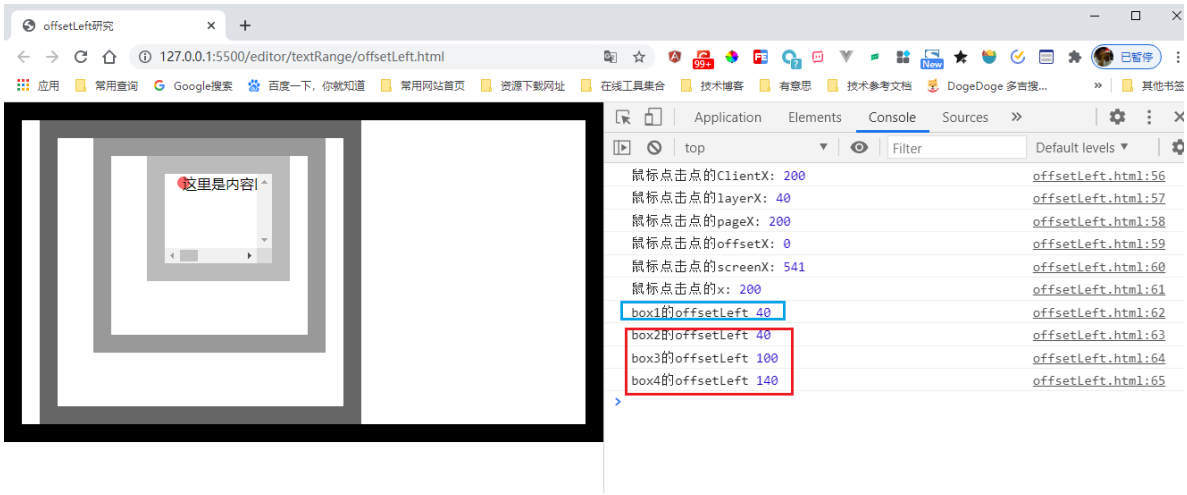
结论1：当元素的祖先元素中没有定位元素时，`offsetLeft`就是元素相对于其`offsetParent`所指向的元素（`body`）的左边框的左侧到元素左边框左侧的距离；

现在就有疑问了，根据`offsetParent`的特征，第一张图那里`offsetParent`明显也是`body`元素，为什么它定位和不定位会有包含不包含`margin`值的区别呢？

body取消定位，box1定位

```
<body style="border: 20px solid #000; margin:0;">
    <div id="box1" style="border: 20px solid #666; width:300px; height:300px; padding-left:20px; margin-left:20px; position: relative;">
```

我们取消`body`元素的定位，给`box1`定位，此时，根据`offsetParent`的规则，`box1/box2/box3`的`offsetParent`应该是`box1`，我们点击相同位置，看输出结果：



此时，这些值都发生了变化：

- `box1` 当前的 `offsetParent` 还是 `body` 元素，所以其 `offsetLeft` 还是它的左边框左侧到 `body` 左边框左侧的距离（40），这个距离包含它自己的 `margin-left` 值。
- `box2` 的 `offsetParent` 是 `box1`，所以它的 `offsetLeft` 值是它的左边框左侧到 `box1` 左边框左侧的距离（60）减去它自己的 `margin-left` 值（20）之后的值（40）。
- `box3` 的 `offsetParent` 是 `box1`，所以它的 `offsetLeft` 值是它的左边框左侧到 `box1` 左边框左侧的距离（120）减去它自己的 `margin-left` 值（20）之后的值（100）。
- `box4` 的 `offsetParent` 是 `box1`，所以它的 `offsetLeft` 是它的左边框左侧到 `box1` 左边框左侧的距离（160）减去它自己的 `margin-left` 值（20）之后的值（140）。

我们得到了

结论3：当祖先元素中有定位元素时，元素的 `offsetLeft` 的值等于它的左边框左侧到它的 `offsetParent` 元素左边框的距离减去它自身的 `margin-left` 值。

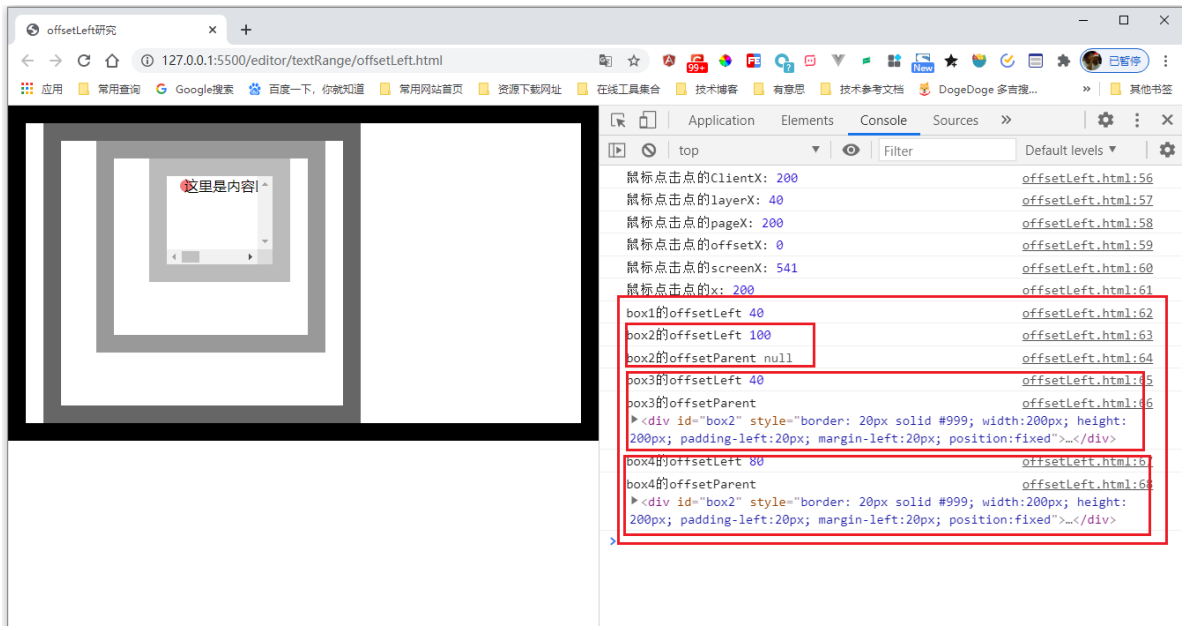
元素自身为fixed定位

由于元素自身为 `fixed` 定位时，它的 `offsetParent` 总是为 `null`，那么作为一个基于 `offsetParent` 的偏移量，元素的 `offsetLeft` 的结果又将是什么呢？

我们修改代码，在上一个例子的基础上，将 `box2` 的定位设为 `fiexd`：

```
1 <body style="border: 20px solid #000; margin:0;">
2     <div id="box1" style="border: 20px solid #666; width:300px; height:300px;
  padding-left:20px; margin-left:20px; position: relative;">
3         <div id="box2" style="border: 20px solid #999; width:200px; height: 200px;
  padding-left:20px; margin-left:20px; position:fixed">
4
```

再次实验：



这次的结果变化比较有意思，我们逐一分析:

- 此时 `box1` 的 `offsetParent` 仍旧是 `body` 元素，所以其 `offsetLeft` 还是它的左边框左侧到 `body` 左边框左侧的距离（40），这个距离包含它自己的 `margin-left` 值。
- `box2` 由于其自身是 `fiexd` 定位，我们可以看到它的 `offsetParent` 为 `null`，可是它的 `offsetLeft` 却并不为 `null` 而是其左边框左侧到 `body` 左边框左侧的距离，这个距离包含它自己的 `margin-left` 值。
- `box3` 的 `offsetParent` 此时变成离它最近的定位祖先元素 `box2`，它的 `offsetLeft` 值则是它自己的左边框左侧到

  `box2` 的左边框左侧的距离，减去它自己的 `margin-left` 值。
- `box4` 的 `offsetParent` 此时也变成了 `box2`，它的 `offsetLeft` 值是它自己的左边框左侧到 `box2` 的左边框左侧的距离，减去它自己的 `margin-left` 值。

我们得到

结论4：当元素自身为 `fixed` 定位时，它的 `offsetLeft` 值等于它自己的左边框左侧到 `body` 左边框左侧的距离，这个距离包含其自身的 `margin-left` 值

元素自身隐藏

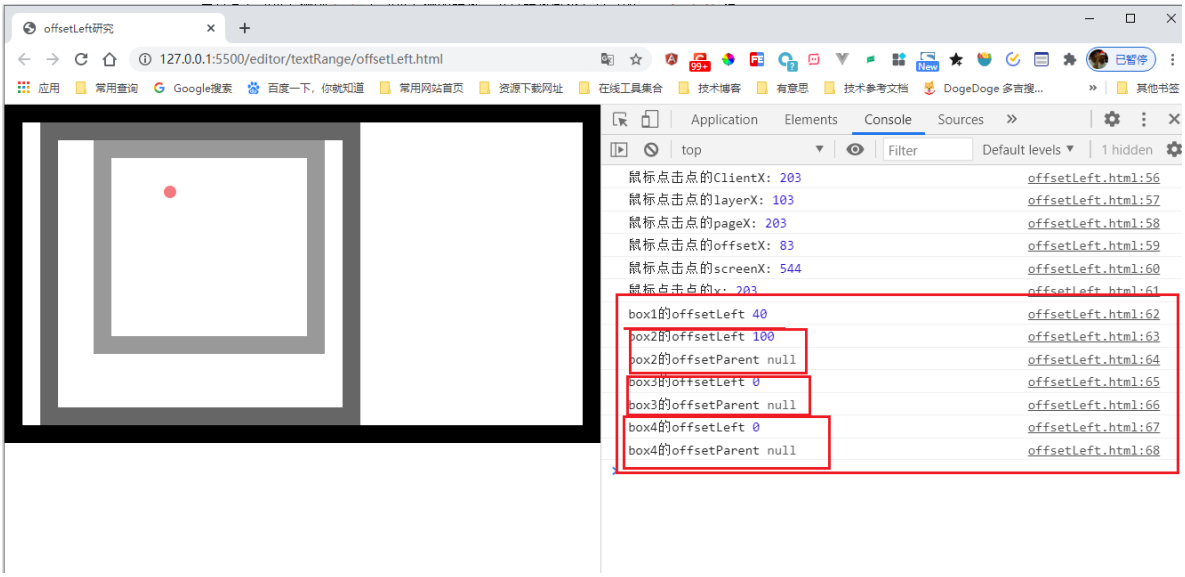
根据 `offsetParent` 的规则，当元素自身的 `display` 为 `none` 时， 它的 `offsetParnt` 是 `null` ，那么它的 `offsetLeft` 会不会发生变化呢？

我们在上一个例子的基础上将 `box3` 的 `display` 设置为 `none` :

```
<body style="border: 20px solid #000; margin:0;">
    <div id="box1" style="border: 20px solid #666; width:300px; height:300px; padding-left:20px; margin-left:20px; position: relative;">
        <div id="box2" style="border: 20px solid #999; width:200px; height: 200px; padding-left:20px; margin-left:20px; position: relative;">
            <div id="box3" style="border: 20px solid #bbb;width:100px; height: 100px; padding-left:20px; margin-left: 20px; overflow:scroll; white-space: nowrap; display: none;">

```

此时由于 `box3` 设置为了隐藏，所以 `box3` 和 `box4` 在页面上不可见了，我们无法再点击 `box4` 来进行试验，但是其实根据我们上面的试验，也同时发现，无论鼠标点击哪里，元素的 `offsetLeft` 值不会随着鼠标点击位置的不同而变化，所以我们就随意点击页面了，看下结果：



观察后发现：

- `box1` 与 `box2` 的 `offsetLeft` 仍旧遵循之前的规则
- `box3` 与 `box4` 的 `offsetParent` 为 `null` ，而他们的 `offsetLeft` 为0

我们得到

结论5：当元素自身 `display` 为 `none` 时，它的 `offsetLeft` 值为 `0`

至此，关于 `offsetLeft` ，我们得出了以下结论：

- 当元素的祖先元素中没有定位元素时，它的 `offsetLeft` 值等于它自己的左边框左侧到 `body` 左边框左侧的距离，这个距离包含其自身的 `margin-left` 值。
- 当祖先元素中有定位元素时，元素的 `offsetLeft` 的值等于它的左边框左侧到它的 `offsetParent` 元素左边框的距离减去它自身的 `margin-left` 值。
- 当元素自身为 `fixed` 定位时，它的 `offsetLeft` 值等于它自己的左边框左侧到 `body` 左边框左侧的距离，这个距离包含其自身的 `margin-left` 值。
- 当元素自身 `display` 为 `none` 时，它的 `offsetLeft` 值为 `0`
- 滚动区域内的内容滚动后，内容所在元素的 `offsetLeft` 值不受影响。