DOM 押题

请简述 DOM 事件模型

先经历从上到下的捕获阶段，再经历从下到上的冒泡阶段。

addEventListener('click',fn,true/false) 第三个参数可以选择阶段。

可以使用 `event.stopPropagation()` 来阻止捕获或冒泡。

手写事件委托

错误版（但是可能能通过面试）

```
ul.addEventListener('click', function(e){
    if(e.target.tagName.toLowerCase() === 'li'){
        fn()// 执行某个函数
    }
})
```

bug 在于，如果用户点击的是 li 里面的 span，就没法触发 fn，这显然不对。

好处

1. 节省监听器

2. 实现动态监听

坏处

调试比较复杂，不容易确定监听者。

解决坏处

解决不了

高级版（不用背）

思路是点击 span 后，递归遍历 span 的祖先元素看其中有没有 ul 里面的 li。

```
function delegate(element, eventType, selector, fn) {
  element.addEventListener(eventType, e => {
    let el = e.target
    while (!el.matches(selector)) {
      if (element === el) {
        el = null
        break
      }
      el = el.parentNode
    }
    el && fn.call(el, e, el)
  })
  return element
}

delete(ul, 'click', 'li', f1)
```

手写可拖曳 div

参考代码：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>JS Bin</title>
</head>
<body>
  <div id="xxx"></div>
  <script>
    var dragging = false
    var position = null

    xxx.addEventListener('mousedown',function(e){
      dragging = true
      position = [e.clientX, e.clientY]
    })

    document.addEventListener('mousemove', function(e){
      if(dragging === false){return}
      console.log('hi')
      const x = e.clientX
      const y = e.clientY
      const deltaX = x - position[0]
      const deltaY = y - position[1]
      const left = parseInt(xxx.style.left || 0)
      const top = parseInt(xxx.style.top || 0)
      xxx.style.left = left + deltaX + 'px'
      xxx.style.top = top + deltaY + 'px'
      position = [x, y]
    })
    document.addEventListener('mouseup', function(e){
      dragging = false
    })
  </script>
</body>
</html>
```

预览：https://jsbin.com/munuzureya/edit?html,js,output

要点：

1. 注意监听范围，不能只监听 div
2. 不要使用 drag 事件，很难用。
3. 使用 transform 会比 top / left 性能更好，因为可以避免 reflow 和 repaint