13. 节流 vs 防抖，面试怎么考?

1. 核心概念 (Core Concept)

节流（Throttling） 和 **防抖（Debouncing）** 都是用于控制事件在一定时间内触发频率的技术，它们本质上是为了优化性能、减少不必要的计算或DOM操作，尤其是在处理高频事件（如窗口滚动、鼠标移动、键盘输入缩放）时。

2. 为什么需要它? (The "Why")

1. **优化性能：** 防止高频事件在短时间内触发大量回调函数，消耗大量计算资源，导致页面卡顿甚至崩溃。
2. **减少请求：** 对于需要发送网络请求的场景（如输入框搜索建议），避免在用户输入过程中频繁触发请求，减轻服务器压力。
3. **提升用户体验：** 控制某些行为的执行频率，使界面响应更流畅，避免因频繁的视觉更新或操作中断带来的不适。

3. API 与用法 (API & Usage)

节流和防抖通常不是浏览器或JavaScript 原生提供的 API，而是开发者基于 `setTimeout` 或 `requestAnimationFrame` 等机制自行实现的函数或使用第三方库（如 Lodash 的 `_.throttle` 和 `_.debounce`）。

3.1. 节流 (Throttling)

规定在一个周期内，事件最多触发一次。如果事件在周期内再次触发，则忽略。

常见实现思路： 使用一个标志位或时间戳记录上次触发的时间，判断当前触发距离上次触发的时间间隔是否大于等于指定的周期。

```
// 经典基于时间戳的节流实现
function throttle(func, delay) {
  let lastTime = 0; // 记录上次触发的时间

  return function(...args) {
    const context = this;
    const now = Date.now(); // 当前时间

    if (now - lastTime >= delay) {
      // 距离上次触发时间超过延迟，可以执行
      lastTime = now; // 更新上次触发时间
      func.apply(context, args);
    }
    // 否则，什么也不做，忽略这次触发
```

```
  };
}

// 示例用法
window.addEventListener('scroll', throttle(() => {
  console.log('滚动条在滚动 (节流)');
}, 200)); // 每 200ms 最多触发一次
```

3.2. 防抖 (Debouncing)

规定在事件触发后，等待一定时间再执行回调。如果在等待时间内事件再次触发，则重新计时。只有当事件停止触发一段时间后，回调函数才会执行。

常见实现思路： 使用 `setTimeout` 设置一个定时器，如果事件在定时器到期前再次触发，则清除之前的定时器，并重新设置一个新的定时器。

```
// 经典基于 setTimeout 的防抖实现
function debounce(func, delay) {
  let timer = null; // 存储定时器 ID

  return function(...args) {
    const context = this;

    // 如果已有定时器，则清除前一个定时器
    if (timer) {
      clearTimeout(timer);
    }

    // 设置新的定时器
    timer = setTimeout(() => {
      func.apply(context, args);
      timer = null; // 定时器执行完毕后清空
    }, delay);
  };
}

// 示例用法
const inputElement = document.getElementById('myInput');
if (inputElement) {
  inputElement.addEventListener('input', debounce((event) => {
    console.log('输入停止了，执行搜索:', event.target.value);
  }, 500)); // 用户停止输入 500ms 后才执行回调
}
```

4. 关键注意事项 (Key Considerations)

1. **根本区别：**

- **节流：** 保证一定时间内**至少**执行一次（或最多执行一次），周期性执行。
- **防抖：** 保证事件**停止触发后**才执行一次，更适用于行为结束后才需要执行的场景。

2. **应用场景：**
   - **节流：** 适用于需要周期性执行的操作，如滚动加载、缩放手势处理、高频鼠标事件。
   - **防抖：** 适用于只需要在事件停止后执行一次的操作，如搜索框输入、窗口resize结束、拖拽结束。
3. **实现方式：** 节流通常通过时间戳或定时器结合标志位实现；防抖主要通过 `setTimeout` 的清除与重设实现。
4. **面试考察：** 面试中常要求现场实现节流和防抖函数，并能清晰解释两者的区别、实现原理和适用场景。有时会考查更复杂的实现（如立即执行的防抖）。

5. 参考资料 (References)

- Lodash Documentation - `_.throttle`
- Lodash Documentation - `_.debounce`
- MDN Web Docs - `setTimeout`
- MDN Web Docs - `clearTimeout`
- 业界公认的技术博客（如阮一峰的网络日志等，但此处不直接引用特定博客链接，以保证信源的普适性，面试时可提及常见实现的思路即可）