防抖和节流

防抖

在频繁操作的场景下，不需要每次都得到结果，就可以使用防抖来实现。

比如：有一个输入框，用户输入一些关键词后，发起 AJAX 请求，返回和这些关键词有关的待选词条。这时候就可以等待用户输入完成暂停后，再等待一段时间才发起请求，避免每输入一个字符就发起请求。

```
<input type="text" id="input" />
```

```
const input = document.getElementById("input");
input.addEventListener("keyup", () => {
    console.log(input.value);
});
```

实现思路：

- 定义一个定时器，先不赋值（或赋值 null）。
- 在逻辑实现函数中，先判断这个定时器有没有值，如果有就清除这个定时器。
- 然后将 setTimeout 的返回值赋值给这个定时器，并等待一定时间再执行逻辑。每次执行完逻辑都需要将定时再赋值 null，保证下一次用户操作再走一遍清除定时器的逻辑，将上一次的计时重置。

```
const input = document.getElementById("input");
let timer = null;
input.addEventListener("keyup", () => {
    if (timer) {
        clearTimeout(timer);
    }
    timer = setTimeout(() => {
        console.log(input.value);
        timer = null;
    }, 500)
});
```

为了不每一个需要用到的地方都这样写一遍防抖逻辑，需要将其封装为一个方法，需要的地方直接调用。

```
// 封装这个 debounce 函数（方法）

// 传入的 fn 函数表示延迟一定时间后需要执行的逻辑
function debounce(fn, delay = 500) {
    // timer 处于闭包中
    let timer = null;
```

```
    // 函数作为返回值，形成了闭包
    return function() {
        if (timer) {
            clearTimeout(timer);
        }
        timer = setTimeout(() => {
            fn();
            timer = null;
        }, delay)
    }
}
```

节流

在频繁操作的场景下，需要间隔一定时间返回一个结果，就可以使用节流来实现。

比如：拖拽一个元素，实时获取它的位置信息，可以每隔 200ms 返回他的位置信息，没必要每拖动一个 px 都返回。

实现思路：

- 每隔 200s 执行一次这个逻辑。
- 先定义一个值为 null 的定时器。
- 如果这个定时器没值（拖动第一个 px），就指定 200ms 后执行内部逻辑。
- 如果这个定时器有值（拖动的第一个 px 以外的），就直接返回，让上面的逻辑保持有效就行了。

```
.elem {
    width: 100px;
    height: 100px;
    border: 1px solid #000;
    border-radius: 5px;
}
```

```
<div id="elem" class="elem" draggable="true">可拖拽元素</div>
```

```
const elem = document.getElementById("elem");
elem.addEventListener("drag", (e) => {
    console.log(e.offsetX, e.offsetY);
});
```

```
// 实现节流
const elem = document.getElementById("elem");
```

```
let timer = null;
elem.addEventListener("drag", (e) => {
    // 拖动第一 px 时，timer 没值，就会赋值。
    // 拖动第一 px 以外的情况下，400ms 以内的其他拖动，就什么都不做直接返回（这时候 timer 已经有值了）
    // 400ms 以后，就会执行逻辑，并把 timer 重置，并循环上一次逻辑（又等待 400ms 再执行，并把 timer 重置）
    if (timer) {
        return;
    }
    timer = setTimeout(() => {
        console.log(e.offsetX, e.offsetY);
        timer = null;
    }, 400);
});
```