throttling and debouncing in javascript

Debouncing and throttling are techniques in JavaScript used to optimize performance by controlling how often a function executes, particularly in response to high-frequency events.

Debouncing

Debouncing ensures that a function is executed only after a specified period of inactivity following a series of events. If a new event occurs within that period, the previous timer is reset, and the function's execution is delayed again. This means the function only fires once, after the user has stopped triggering the event for a certain duration.

Use Cases: Search bar input (making an API call only after the user stops typing), window resize events (updating layout only after resizing is complete).
Example Implementation:

JavaScript

```
function debounce(func, delay) {
    let timeout;
    return function(...args) {
        const context = this;
        clearTimeout(timeout);
        timeout = setTimeout(() => func.apply(context, args), delay);
    };
}

const debouncedSearch = debounce(() => console.log("Performing search..."), 500);
// Attach debouncedSearch to an input's 'keyup' event
```

Throttling

Throttling limits the rate at which a function can be executed. It ensures that a function is called at most once within a specified time interval, regardless of how many times the event is triggered within that interval.

Use Cases: Scroll events (updating UI elements periodically during scrolling), mouse movement tracking, button clicks with rate limits.
Example Implementation:

JavaScript

```
function throttle(func, limit) {
    let inThrottle;
    return function(...args) {
        const context = this;
        if (!inThrottle) {
            func.apply(context, args);
            inThrottle = true;
            setTimeout(() => (inThrottle = false), limit);
        }
    };
}

const throttledScrollHandler = throttle(() => console.log("Scrolling..."), 200);
// Attach throttledScrollHandler to a 'scroll' event
```

Key Differences

Debouncing executes once after a period of inactivity, while throttling executes at a fixed rate within an interval.

Debouncing consolidates multiple rapid events into a single action, while throttling ensures a function doesn't execute too frequently.

Revision #2
Created 29 October 2025 02:43:38 by AI API
Updated 5 December 2025 09:28:51 by AI Channel