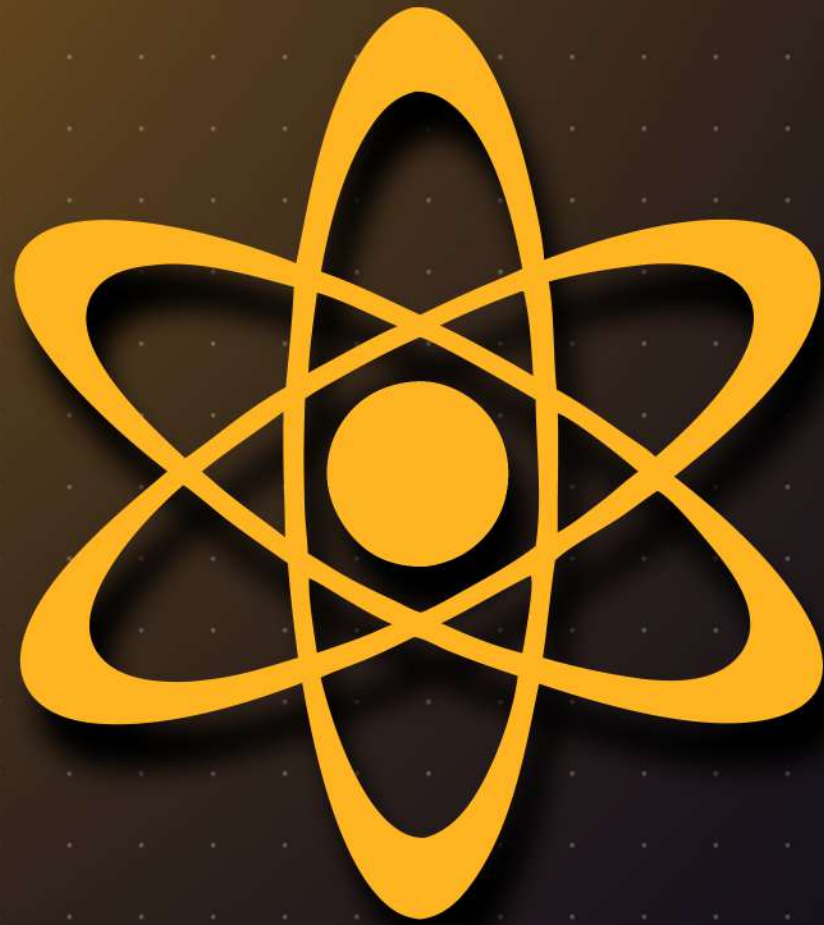


Debouncing vs Throttling:



Understanding the Differences for Performance Optimization



SUMANTH M
Frontend developer



1

What is Debouncing?



- Debouncing is a technique that ensures a function is only executed after a specified time has elapsed since the last time it was invoked.
- Ideal for scenarios where you want to execute a function only after the user has stopped an action.
- **Commonly used in:**
 - Search input fields (to limit API calls while typing)
 - Window resizing events



SUMANTH M
Frontend developer



2

What is Throttling?



- Throttling is a technique that ensures a function is executed at most once in a specified time interval.
- It allows a function to run repeatedly at defined intervals, regardless of how many times the event is triggered.
- **Commonly used in:**
 - Scroll events (for infinite scrolling)
 - Resize events



SUMANTH M
Frontend developer



How Does Debouncing Work?

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

Debouncing waits for a pause in events before executing the function.

If the event fires again within the specified time, the timer resets.

Key Point: The function is executed only after the specified delay, making it ideal for scenarios like search inputs.



SUMANTH M
Frontend developer



How Does Throttling Work?

```
function throttle(func, limit) {  
  let lastFunc;  
  let lastRan;  
  
  return function(...args) {  
    if (!lastRan) {  
      func.apply(this, args);  
      lastRan = Date.now();  
    } else {  
      clearTimeout(lastFunc);  
      lastFunc = setTimeout(() => {  
        if (Date.now() - lastRan >= limit) {  
          func.apply(this, args);  
          lastRan = Date.now();  
        }  
      }, limit - (Date.now() - lastRan));  
    }  
  };  
}
```

Throttling allows a function to execute at regular intervals, no matter how many times the event is triggered.

Key Point: The function is executed at most once every specified interval, making it suitable for high-frequency events like scrolling.



Key Differences



Feature Debouncing Throttling

Execution Timing Executes after a pause

Executes at regular intervals

Use Case Ideal for events triggered by user input

Ideal for events that fire rapidly

Example Search input, form validation

Scroll events, window resizing



SUMANTH M
Frontend developer



When to Use Debouncing?



- **Search Inputs:** Limit API calls while the user types.
- **Window Resizing:** Reduce the frequency of resize event handling.
- **Form Submissions:** Prevent multiple submissions during user input.



SUMANTH M
Frontend developer



When to Use Throttling?



- **Scroll Events:** Manage performance during heavy scroll activity.
- **Resize Events:** Optimize performance while resizing the window.
- **Button Clicks:** Prevent multiple clicks on buttons, especially during network requests.



SUMANTH M
Frontend developer



8

Performance Benefits



- **Debouncing:** Reduces the number of function calls by executing only after the user has finished an action, enhancing resource management.
- **Throttling:** Controls the rate of function execution, preventing performance bottlenecks during
- rapid event firing.



SUMANTH M
Frontend developer



Wrap-Up

Understanding when to use debouncing or throttling is crucial for optimizing performance in web applications. Both techniques help manage resource usage effectively, but they serve different purposes based on the use case.



SUMANTH M
Frontend developer

