Somyaranjan Sethy

Frontend / React / UI

🕒 Debounce vs Throttling: What's the Difference? 🤔

Save Post

Swipe To Know

Somyaranjan Sethy

Frontend / React / UI

🕒 Debounce :

- **Purpose:** Delays invoking a function until after a specified time period has elapsed since the last time the function was invoked.
- **Use Case:** Preventing frequent API requests on user input.

🕒 Throttling:

- **Purpose:** Limits the rate at which a function can fire, typically to a fixed interval.
- **Use Case:** Optimizing scroll event handlers to execute at a controlled frequency.

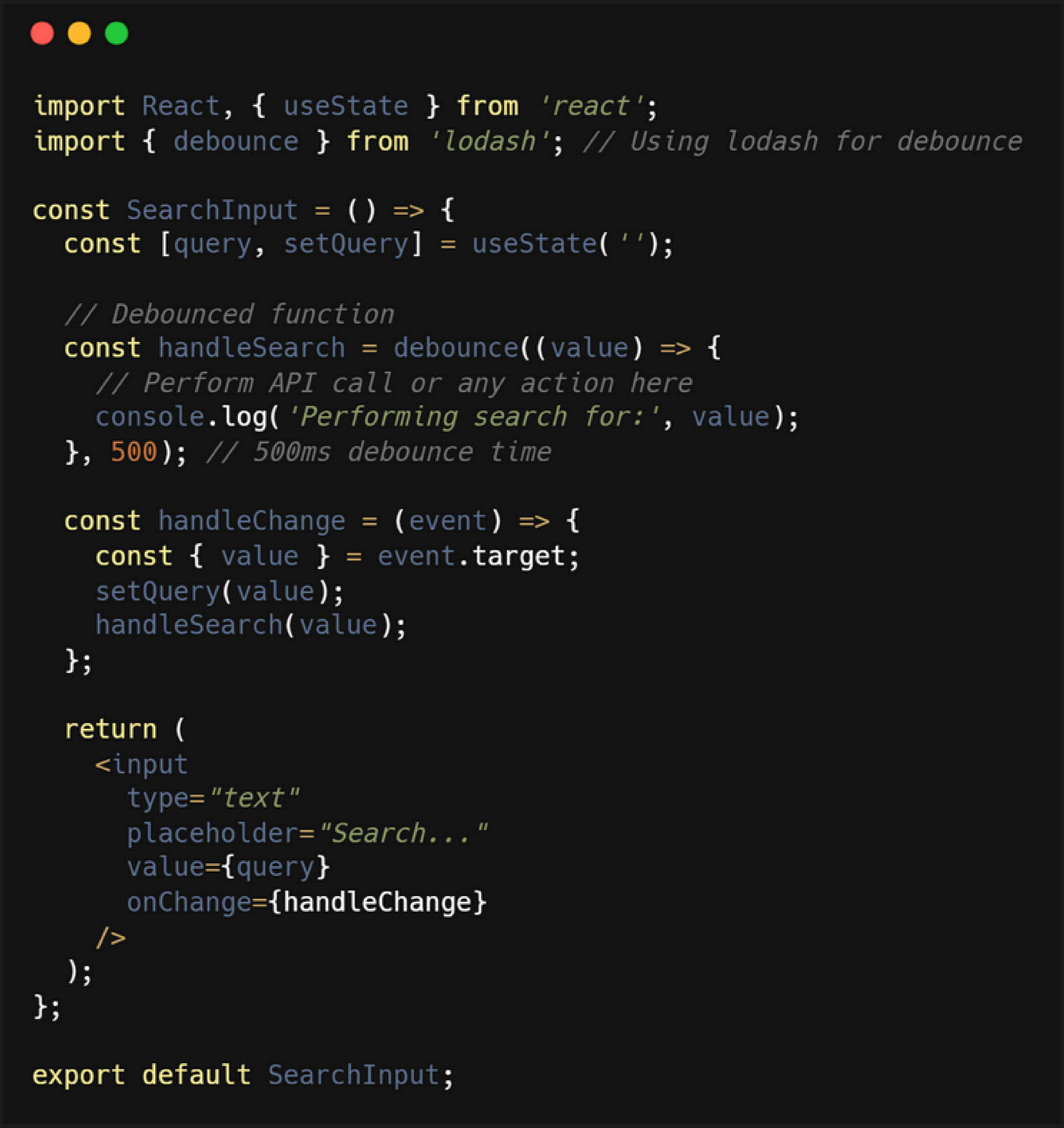
Save Post

Swipe To Know

Somyaranjan Sethy

Frontend / React / UI

</> Debounce Code Sinppet:



```
import React, { useState } from 'react';
import { debounce } from 'lodash'; // Using lodash for debounce

const SearchInput = () => {
  const [query, setQuery] = useState('');

  // Debounced function
  const handleSearch = debounce((value) => {
    // Perform API call or any action here
    console.log('Performing search for:', value);
  }, 500); // 500ms debounce time

  const handleChange = (event) => {
    const { value } = event.target;
    setQuery(value);
    handleSearch(value);
  };

  return (
    <input
      type="text"
      placeholder="Search..."
      value={query}
      onChange={handleChange}
    />
  );
};

export default SearchInput;
```

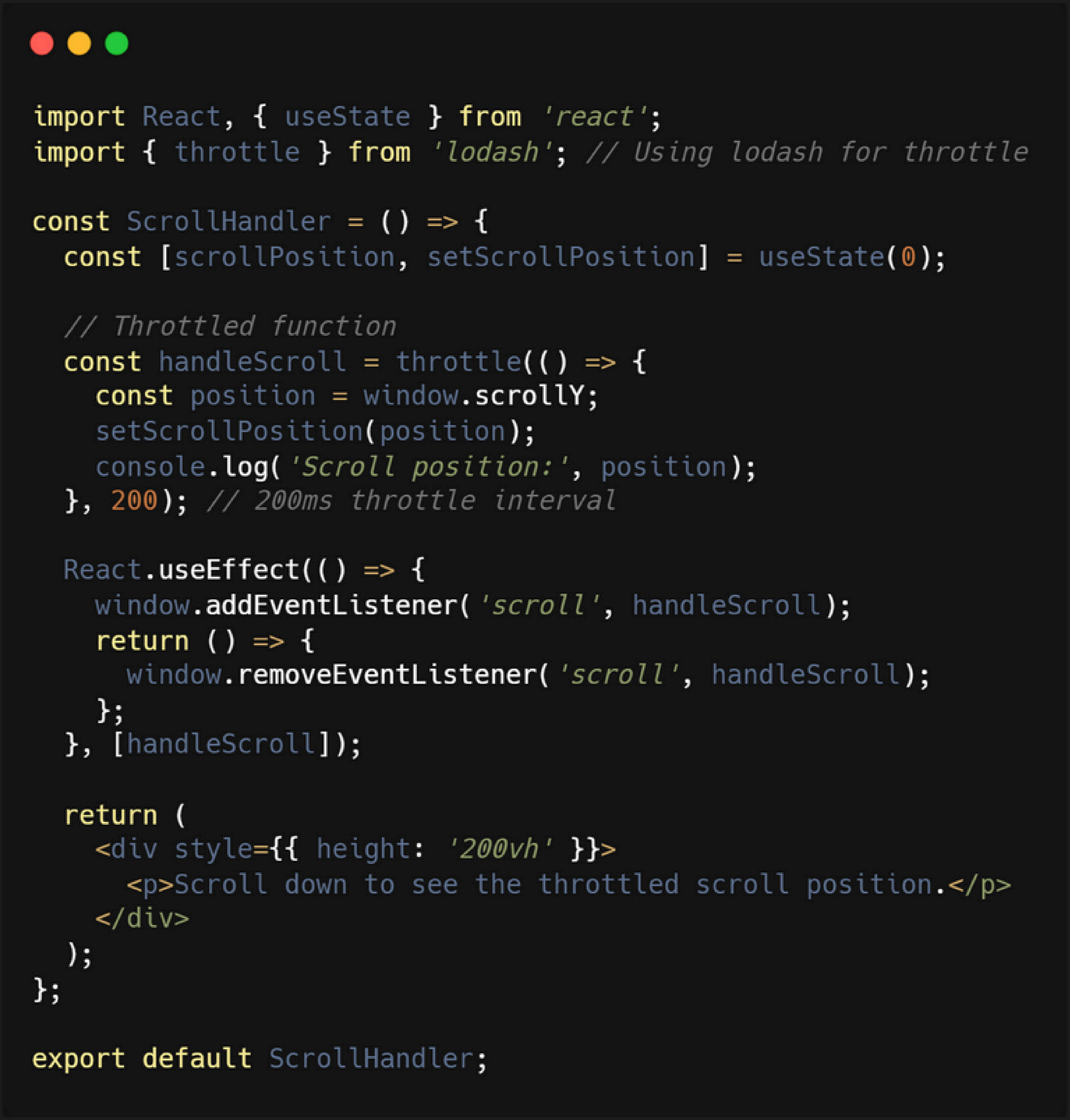
Save Post

Swipe To Know

Somyaranjan Sethy

Frontend / React / UI

</> Throttling Code Sinppet:



```
import React, { useState } from 'react';
import { throttle } from 'lodash'; // Using lodash for throttle

const ScrollHandler = () => {
  const [scrollPosition, setScrollPosition] = useState(0);

  // Throttled function
  const handleScroll = throttle(() => {
    const position = window.scrollY;
    setScrollPosition(position);
    console.log('Scroll position:', position);
  }, 200); // 200ms throttle interval

  React.useEffect(() => {
    window.addEventListener('scroll', handleScroll);
    return () => {
      window.removeEventListener('scroll', handleScroll);
    };
  }, [handleScroll]);

  return (
    <div style={{ height: '200vh' }}>
      <p>Scroll down to see the throttled scroll position.</p>
    </div>
  );
};

export default ScrollHandler;
```

Save Post

Swipe To Know

Somyaranjan Sethy

Frontend / React / UI

⚖️ Key Differences :

- **Debounce:** Delays execution of a function until after a specified time since the last call.
- **Throttling:** Limits the execution rate of a function to a fixed interval.
- **Use Cases:** Debounce for input events to reduce frequency, throttling for scroll or resize events to control execution frequency.

Save Post

Swipe To Know

Somyaranjan Sethy

Frontend / React / UI

🎯 Conclusion

Understanding when to use debounce or throttling can significantly improve your React.js applications' performance and user experience. Choose wisely based on your specific use case!

🛠️ Keep Learning!

There's always more to explore in frontend development. Share your thoughts and experiences with debounce and throttling in the comments! Let's continue learning together. 💬🌟

Save Post