Debouncing & Throttling

Debouncing	Throttling
Debouncing waits for a certain time before invoking the function again	Throttling limits the number of times the function can be called over a certain period
Ensures that the function is called only once, even if the event is triggered multiple times	Ensures that the function is called at a regular interval, even if the event is triggered multiple times
Useful when you want to delay the invocation of a function until a certain period of inactivity has passed	Useful when you want to limit the frequency of function calls
Eg. You can debounce an async API request function that is called every time the user types in an input field.	Eg. You can throttle a slide change function that is called every time the user clicks a button in a carousel.

Debouncing

Debouncing is a technique where you delay the execution of a function until after a certain amount of time has passed since the last time it was called.

This is useful when you have a function that gets called frequently, such as an event listener for scroll or resize events, and you want to avoid triggering it too often and bogging down the browser.

Debouncing

For example, you can debounce an asynchronous API request function that is called every time the user types in an input field. Without debouncing, this would send a new request for every keystroke, which can be inefficient and slow down your web app.

Debouncing

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

const debouncedFunction = debounce(function() {
  console.log('Function called after 500ms of inactivity');
}, 500);

window.addEventListener('scroll', debouncedFunction);
```

Throttling

Throttling is a similar technique to debouncing, but instead of delaying the execution of a function, it limits the rate at which a function can be called.

This is useful when you have a function that can be called frequently but doesn't need to be executed every time, such as an event listener for mousemove or keydown events.

Throttling

For example, you can throttle a slide change function that is called every time the user clicks a button in a carousel. Without throttling, the user can spam-click the buttons and cause the carousel to glitch or freeze.

```javascript
function throttle(func, limit) {
  let timerId;
  let lastTime = 0;
  return function() {
    const context = this;
    const args = arguments;
    const now = Date.now();
    if (now - lastTime >= limit) {
      func.apply(context, args);
      lastTime = now;
    } else {
      clearTimeout(timerId);
      timerId = setTimeout(() => {
        func.apply(context, args);
        lastTime = Date.now();
      }, limit - (now - lastTime));
    }
  }
}

const throttledFunction = throttle(function() {
  console.log('Function called at most once every 100ms');
}, 100);

window.addEventListener('mousemove', throttledFunction);
```

That' all folks 👏

That's it for today's post on debouncing and throttling in JavaScript! I hope this has been helpful in understanding these concepts 🙌