To create a smooth, animated counter for a currency value like **1.200,56€**, you need to handle three things: stripping the formatting to get a raw number, animating the increment, and then re-formatting the number back into the European locale string.
Here is a clean, "legit" implementation using a reusable function.

1. The HTML

We use a data-target attribute to store the final value. This keeps your JavaScript decoupled from the hardcoded text.

HTML

```
<div class="stat-container">
  <span class="count-up" data-target="1200.56">0,00</span><span>€</span>
</div>
```

2. The CSS

Nothing fancy is required for the logic, but we'll add some basic styling to make it look professional.

CSS

```
.stat-container {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  font-size: 3rem;
  font-weight: bold;
  color: #2d3436;
}

.count-up {
  display: inline-block;
}
```

3. The JavaScript

This script uses requestAnimationFrame for high-performance animation and Intl.NumberFormat to ensure the commas and periods are in the correct places for a European format.

JavaScript

```
const animationDuration = 2000; // Total time in milliseconds

const countUp = (el) => {
  const target = parseFloat(el.getAttribute('data-target'));
  const start = 0;
  let startTime = null;

  // Formatter for European style: 1.200,56
  const formatter = new Intl.NumberFormat('de-DE', {
    minimumFractionDigits: 2,
    maximumFractionDigits: 2,
  });

  const step = (currentTime) => {
    if (!startTime) startTime = currentTime;
    const progress = Math.min((currentTime - startTime) / animationDuration, 1);

    // Calculate current value with an "easeOutExpo" timing function
    const currentValue = progress === 1
      ? target
      : target * (1 - Math.pow(2, -10 * progress));

    el.innerText = formatter.format(currentValue);

    if (progress < 1) {
      window.requestAnimationFrame(step);
    }
  };

  window.requestAnimationFrame(step);
};

// Initialize
document.querySelectorAll('.count-up').forEach(countUp);
```

How it works:

- **Performance:** Instead of setInterval, we use requestAnimationFrame. This ensures the animation runs at the refresh rate of your monitor (usually 60fps or 144fps), making it

buttery smooth.

- **Easing:** I included a simple math formula (1 - Math.pow(2, -10 * progress)) so the numbers start fast and slow down as they reach the target. It feels much more premium than a constant speed.
- **Formatting:** Intl.NumberFormat('de-DE') automatically handles the European convention of using a dot for thousands and a comma for decimals.
- **Precision:** Using parseFloat ensures that the decimal values (.56) are calculated accurately during the transition.