Here's a **complete, reusable Angular directive** that tracks **size and position** of an element — including **initial values** and **margins** — using the modern `ResizeObserver` API and `getBoundingClientRect()`.

It works in Angular 9+ and is **fully reactive**.

Directive Code

```
import {
  Directive,
  ElementRef,
  EventEmitter,
  NgZone,
  OnDestroy,
  OnInit,
  Output
} from '@angular/core';

export interface ElementMetrics {
  width: number;
  height: number;
  top: number;
  left: number;
  marginTop: number;
  marginRight: number;
  marginBottom: number;
  marginLeft: number;
}

@Directive({
  selector: '[trackElementMetrics]'
})
export class TrackElementMetricsDirective implements OnInit,
OnDestroy {
  @Output() metricsChange = new EventEmitter<ElementMetrics>();

  private resizeObserver?: ResizeObserver;
```

```
  private mutationObserver?: MutationObserver;

  constructor(private el: ElementRef, private ngZone: NgZone) {}

  ngOnInit(): void {
    // Run outside Angular to avoid unnecessary change detection
cycles
    this.ngZone.runOutsideAngular(() => {
      this.resizeObserver = new ResizeObserver(() =>
this.emitMetrics());
      this.resizeObserver.observe(this.el.nativeElement);

      // MutationObserver to detect DOM changes that might affect
position
      this.mutationObserver = new MutationObserver(() =>
this.emitMetrics());
      this.mutationObserver.observe(document.body, {
        attributes: true,
        childList: true,
        subtree: true
      });

      // Emit initial values
      this.emitMetrics();
    });
  }

  private emitMetrics(): void {
    const rect = this.el.nativeElement.getBoundingClientRect();
    const style = window.getComputedStyle(this.el.nativeElement);

    const metrics: ElementMetrics = {
      width: rect.width,
      height: rect.height,
      top: rect.top + window.scrollY,
      left: rect.left + window.scrollX,
```

```
      marginTop: parseFloat(style.marginTop) || 0,
      marginRight: parseFloat(style.marginRight) || 0,
      marginBottom: parseFloat(style.marginBottom) || 0,
      marginLeft: parseFloat(style.marginLeft) || 0
    };

    // Emit inside Angular zone so subscribers trigger change
detection
    this.ngZone.run(() => this.metricsChange.emit(metrics));
  }

  ngOnDestroy(): void {
    this.resizeObserver?.disconnect();
    this.mutationObserver?.disconnect();
  }
}
```

Usage Example

```
<div
  trackElementMetrics
  (metricsChange)="onMetricsChange($event)"
  style="margin: 20px; width: 200px; height: 100px; background:
lightblue;"
>
  Resize or move me
</div>
```

```
onMetricsChange(metrics: ElementMetrics) {
  console.log('Element metrics:', metrics);
}
```

How It Works

- `ResizeObserver` → Detects changes in element size.

- `MutationObserver` → Detects DOM changes that may affect position.
- `getBoundingClientRect()` → Gets position relative to viewport.
- `window.getComputedStyle()` → Reads margins.
- **Initial Values** → Emitted immediately on `ngOnInit`.
- **Scroll Handling** → Adds `window.scrollX` and `window.scrollY` to get absolute page position.

✅ **Advantages**:

- Works for **both size and position** changes.
- Captures **margins** and **initial values**.
- Runs efficiently outside Angular's zone, only re-entering when emitting data.
- Cleans up observers on destroy.

If you want, I can extend this so it **also tracks scroll events** without relying on `MutationObserver` for better performance.
Do you want me to add that?