EcomLabs

STEP-BY-STEP GUIDE

CLS: a cookie banner that doesn't move the page

How to measure the CLS your cookie banner causes, in Chrome DevTools and with a Console script, the CSS and JavaScript for a banner fixed to the bottom that pushes nothing and doesn't hide keyboard focus, and a checklist to confirm it before and after.

FOR	Any online store with a cookie or consent banner
TESTED WITH	Playwright 1.62.0 · Chromium 151 · axe-core 4.12.1 · Node 22.22.3 · Windows 11
UPDATED	November 11, 2026
DOCS	web.dev/articles/cls
ARTICLE	ecomlabs.dev/g/cls-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What you'll get done
Step 01	Find the shift in Chrome DevTools
Step 02	Measure with a Console script
Step 03	The CSS for a banner that doesn't push
Step 04	The JavaScript that shows it
Step 05	If you'd rather reserve space
Step 06	What we measured
When you're done	Checklist

BEFORE YOU START

What you'll get done

Measure how much your cookie banner moves the page, switch it to a banner fixed to the bottom that appears over the content without pushing anything, and confirm the change. All the code is copied from the test pages we measured locally on November 11, 2026 with Playwright 1.62.0 and Chromium 151. `saveConsent` was a stub: no consent platform was called.

CLS thresholds, at the 75th percentile and separately for mobile and desktop: 0.1 or less is good; above 0.1 up to 0.25 needs improvement; above 0.25 is poor.

STEP 01

Find the shift in Chrome DevTools

Open a product page in an Incognito window so there is no stored consent and the banner shows. Emulate a phone: the same banner gave us 0.1611 at 390 × 844 and 0.0332 at 1366 × 768.

- **Performance:** opening the panel shows your local LCP and CLS and, in the **Layout shifts** tab, a table of shifts.
- **Record and reload:** check **Screenshots** and record. DevTools goes through `about:blank` and records the full load.
- **Layout shifts track:** each purple diamond is a shift. Click it: **Summary** shows timings, score, elements and potential culprits.
- **Insights › Layout shift culprits:** its best guess at the cause (unsized images, injected iframes, animations, fonts).
- **Rendering › Layout Shift Regions:** press `Ctrl+Shift+P`, type `rendering`, pick **Show Rendering** and check the box. On reload, whatever moves flashes purple.

STEP 02

Measure with a Console script

Paste it into the Console once the banner has appeared; `buffered: true` picks up earlier shifts:

```
let cls = 0, windowValue = 0, windowStart = 0, lastShift = -Infinity;

new PerformanceObserver((list) => {
  for (const entry of list.getEntries()) {
    if (entry.hadRecentInput) continue;
    const t = entry.startTime;
    if (t - lastShift >= 1000 || t - windowStart >= 5000) {
      windowValue = 0;
      windowStart = t;
    }
    windowValue += entry.value;
    lastShift = t;
    cls = Math.max(cls, windowValue);
```

```
    console.log("shift", entry.value.toFixed(4), "CLS", cls.toFixed(4),
      entry.sources.map((s) => s.node));
  }
}).observe({ type: "layout-shift", buffered: true });
```

It drops shifts within 500 ms of a tap, click or keypress (`hadRecentInput`), applies the session windows and keeps the worst one: that is the CLS. `sources` lists the elements that moved, not the cause: with the banner on top you get the header and main. MDN marks the LayoutShift API as experimental and not Baseline, so measure in Chrome.

STEP 03

The CSS for a banner that doesn't push

Fixed to the bottom and out of the flow, with the page's spacing matched to its height:

```
.cookie-banner {
  position: fixed;
  inset: auto 0 0 0;
  z-index: 100;
  display: flex;
  flex-wrap: wrap;
  align-items: center;
  gap: 0.75rem 1.5rem;
  max-height: 50vh;
  overflow-y: auto;
  padding: 1rem 1rem calc(1rem + env(safe-area-inset-bottom));
  background: #14171a;
  color: #f4f5f6;
}
.cookie-banner p { flex: 1 1 32ch; margin: 0; }
.cookie-banner a { color: inherit; }
.cookie-banner .actions { display: flex; gap: 0.5rem; }
.cookie-banner button {
  min-height: 44px;
  padding: 0 1.25rem;
  font: inherit;
  color: inherit;
  background: transparent;
  border: 1px solid currentColor;
  border-radius: 4px;
}
.cookie-banner :focus-visible { outline: 3px solid #ffd166; outline-offset: 2px; }
html { scroll-padding-bottom: var(--cookie-banner-h, 0px); }
body { padding-bottom: var(--cookie-banner-h, 0px); }
@media (prefers-reduced-motion: no-preference) {
  .cookie-banner { animation: cookie-banner-in 200ms ease-out; }
}
@keyframes cookie-banner-in { from { transform: translateY(100%); } }
```

- **No shifts:** `position: fixed` places it over the content, and the entrance uses `transform`, which doesn't count as a layout shift.
- **Focus stays visible:** `scroll-padding-bottom` keeps the focused element from ending up underneath (WCAG 2.2 criterion 2.4.11, technique C43); `padding-bottom` lets the end of the page scroll into view.
- **Contrast:** text at 16.48:1 and a focus ring at 12.48:1 against the banner background; 44 px buttons.
- **Mobile:** `max-height: 50vh` with internal scrolling, so it never covers more than half the screen.

STEP 04

The JavaScript that shows it

Call `showCookieBanner(banner)` from your consent script with the banner element already built:

```
function showCookieBanner(banner) {
  const root = document.documentElement;
  const observer = new ResizeObserver(() => {
    root.style.setProperty("--cookie-banner-h", banner.offsetHeight + "px");
  });
  document.body.prepend(banner);
  observer.observe(banner);
  banner.addEventListener("click", (event) => {
    const button = event.target.closest("button[data-consent]");
    if (!button) return;
    saveConsent(button.dataset.consent);
    observer.disconnect();
    banner.remove();
    root.style.removeProperty("--cookie-banner-h");
  });
}
```

The banner markup we used in the test:

```
<section class="cookie-banner" aria-label="Cookies">
  <p>We use cookies to measure visits and remember your cart.
    <a href="/privacy">Privacy policy</a></p>
  <div class="actions">
    <button type="button" data-consent="reject">Reject</button>
    <button type="button" data-consent="accept">Accept</button>
  </div>
</section>
```

`prepend` makes it first in tab order without moving anything, `ResizeObserver` keeps the height current when the phone rotates, and `saveConsent` is yours: your consent platform or your own cookie.

> **CAUTION**
>
> If your theme has a sticky Add to cart bar, the banner covers it. Lift it with `translate: 0 calc(-1 * var(--cookie-banner-h, 0px))`; changing its `bottom` logged a 0.0187 shift.

STEP 05

If you'd rather reserve space

This only works when the server knows whether consent exists and the banner's height is stable: the server prints a `<div class="cookie-slot">` with a `min-height` per breakpoint only when the request has no consent cookie, and the script puts the banner inside it. Measure your banner at every width: ours is 136 px tall up to 480 px and 76 px from 767; a single 80 px slot gave 0.0601 on mobile, and a slot a script removes later gave 0.1611.

STEP 06

What we measured

Measured locally on a test page: each variant served from memory by Playwright, Chromium 151, three identical runs per variant.

VARIANT	MOBILE	DESKTOP
Banner on top, in the flow	0.1611	0.0332
Banner fixed to the bottom (this CSS)	0	0
Fixed banner animating `bottom`	0.0143	0.0031
Image without `width` / `height`	0.1835	0.0862
Image with `width` / `height`	0	0

These are lab numbers. What counts for Core Web Vitals is field data: PageSpeed Insights shows a 28-day average of real users, so your fix takes weeks to show in full.

WHEN YOU'RE DONE

Checklist

- [ ] You measured CLS in Incognito, with mobile emulation and on desktop, and wrote the value down.
- [ ] The banner is out of the flow: `position: fixed` at the bottom, or in a slot the server prints.
- [ ] The entrance uses `transform` and doesn't run under `prefers-reduced-motion: reduce`.
- [ ] `scroll-padding-bottom` and `padding-bottom` use the banner's real height, and no focused link ends up under it when you tab.
- [ ] It covers no more than half the screen and doesn't hide the sticky buy bar on mobile.
- [ ] Text contrast is 4.5:1 or more, focus is visible, Accept and Reject are the same size.
- [ ] Every image has `width` and `height`; ads and iframes get a `min-height`.
- [ ] You measured again, the banner's arrival logs no shift, and you check the field data over the following weeks.