Support target-counter() and target-counters() from CSS Generated Content Module

(public version)

This Document is Public Set doc permissions to: "Anyone with the link" Add contributors@chromium.org as either editors or commenters (uncheck "notify" when doing so)

Authors: jontyleslie@gmail.com

July 2026

One-page overview

Summary

Implement target-counter() and target-counters() from the CSS Generated Content Module Level 3 content property, allowing authors to render the value of a CSS counter at a link's target element, in place at the link's location. This is most commonly used for page-number cross-references and tables of contents in print/paged media (e.g. "see page 137", "Chapter One.....1"). target-text() is being tracked as a separate, follow-up design doc since it has a substantially different implementation surface (text/accessibility content extraction rather than counter resolution).

Platforms

Mac, Windows, Linux, Chrome OS, Android, Android WebView, Fuchsia. (iOS uses WebKit). The feature is primarily exercised via @media print / window.print() and headless printing (e.g. --headless --print-to-pdf), since pagination, and thus a meaningful target *page*, only exists in paged contexts.

Team

jontyleslie@gmail.com, first-time external contributor. Full-time software developer at RMHEDGE, Melbourne. Seeking an OWNER-level sponsor/reviewer from Blink>CSS or Blink>Layout>Printing (see Communicate, below).

Bug

crbug.com/40529223, "Support cross-references from CSS Generated Content for Paged Media Module." Open since July 2017, confirmed (P3/Available), with a steady stream of "+1, please" comments through 2026. Its sole blocker, crbug.com/40529222 ("Support page-based counters from CSS Paged Media Module Level 3"), is **Fixed**, so the page-aware counter resolution this feature depends on already exists in Blink.

Code affected

- **Blink>CSS**: parser/value support for the target-counter() / target-counters() functional notations as new <content-list> value types.
- **Blink>Layout>Printing** / pagination: resolving a counter's value at the *target* element's position in the paginated layout tree, which may be on a different page than the reference.
- **Blink>CSS>Counters**: reusing/extending the existing counter resolution machinery (landed for crbug.com/40529222) to support "resolve as of this other node" rather than only "resolve as of self."
- **Style engine / invalidation**: target-counter() results must invalidate and re-resolve when (a) the referenced element's counter value changes, or (b) pagination shifts which page the target lands on, since the formatted output (e.g. "page 12" vs "page 13") depends on layout, not just style.

Design

Background and motivation

Authors writing long-form printed/PDF documents, technical manuals, books, legal contracts, generated reports, routinely need "see page N" cross-references and tables of contents with page numbers. Today, Chromium-based browsers/print-to-PDF pipelines have no way to do this in CSS. Authors must either omit page references entirely or use JavaScript print-time hacks that don't survive pagination changes. This is one of the most-requested missing pieces of CSS Paged Media support, evidenced by 22 comments spanning nine years on the tracking bug, including repeated unprompted "+1" comments from unrelated parties as recently as February 2026.

The CSSWG's own canonical example is target-counter():

a::after { content: target-counter(attr(href url), page) }

"…which will be discussed on page 137."

and the table-of-contents case:

.frontmatter a::after { content: leader('.') target-counter(attr(href url), page, lower-roman) }
.bodymatter a::after { content: leader('.') target-counter(attr(href url), page, decimal) }

Preface.....vii Chapter One.....1

Spec status note

The CSS Generated Content Module Level 3 is a W3C Working Draft and is explicitly marked "a very rough draft, and is not ready for implementation" at the top of the document. Several normative details are unresolved as open spec issues, most relevantly:

- **Error handling** for target-counter() when the target URL has no fragment, the ID doesn't exist, or it points outside the current document. The spec currently only says "the user agent must treat that as an error" without defining what that means observably (issue #7443dd19 in the spec).
- A note restricting the feature to **same-document fragment URLs only** for now (issue #2b2bd98b). This actually simplifies our job, since we do not need to support cross-document references.

This design therefore proposes Chromium-specific, spec-compatible interim behavior for the underdefined cases (see Error Handling, below), flagged clearly as such,

consistent with the contributing guide's expectation that nontrivial features get reviewed by people familiar with the area before implementation proceeds.

Scope of this document

In scope:

- target-counter(<url>, <counter-name>, <counter-style>?)
- target-counters(<url>, <counter-name>, <string>, <counter-style>?) (shares the bulk of its implementation with target-counter(): same target-resolution and counter-lookup logic, differing only in formatting multiple nested counter values rather than just the innermost one)
- Restricting <url> to local fragment references (<a href="#foo">-style), per the spec's own recommended restriction.
- Behavior in paginated/print contexts (@media print, window.print(), headless print-to-PDF).

Explicitly out of scope (tracked separately):

- target-text(): different implementation surface (extracting rendered text/accessibility content of the target rather than a counter value). Will be proposed in a follow-up design doc.
- Cross-document fragment references: deferred per spec guidance. Would need a security/privacy review of its own (leaking information about whether/where a cross-origin document has a given ID).
- Non-paged-media use of these functions (the spec technically allows target-counter() outside paged contexts, but it is meaningless without a page counter or other page-relative context. Chromium will support it generally but most real usage is print/paged).

Implementation details

1. Parsing. Add target-counter() and target-counters() as new functional-notation members of <content-list>/<target> in the CSS value parser (CSSParserFastPaths / the relevant css_parsing_utils grammar for content). Argument grammar:

target-counter() = target-counter([<string> | <url>], <custom-ident>, <counter-style>?)
target-counters() = target-counters([<string> | <url>], <custom-ident>, <string>, <counter-style>?)

This produces a new CSSTargetCounterValue (or similar) CSSValue subtype, analogous to the existing CSSCounterValue used for plain counter()/counters().

2. Resolving the target element. At style/layout resolution time, resolve the <url> argument (typically attr(href url)) to a fragment identifier, then look up the

corresponding element by ID within the current document, reusing existing fragment-navigation lookup code rather than introducing new ID-resolution logic.

3. Resolving the counter value at the target. This is the core of the feature and the reason the page-counters blocker mattered. We need the value of the named counter as it stood at the target element's position in the document, not at the position of the element containing target-counter(). This means walking the existing counter-tree machinery (built for counter()/counters() and extended for page counters under crbug.com/40529222) anchored at the target node instead of the calling node. This is the main net-new logic in this CL: today's counter resolution API assumes "resolve relative to the current node in normal tree order"; we need a "resolve relative to an arbitrary other node" entry point.

4. Layout/pagination dependency. The rendered value (e.g. the page counter) depends on which page the target element ends up on, and pagination itself depends on layout that hasn't necessarily happened yet for forward references. Because of this, target-counter() results must be treated as **layout-dependent generated content**: resolved (or re-resolved) once pagination is complete, with invalidation if a later layout pass moves the target to a different page (e.g. due to content reflow). This mirrors how string(...) / running headers already depend on page layout in existing paged-media support. We'll align with that existing invalidation pathway rather than building a new one.

5. Formatting. Once the counter value is resolved, formatting via the optional <counter-style> argument reuses the existing @counter-style / predefined counter-style formatting code already used by counter().

6. target-counters() specifics. Differs from target-counter() only in: (a) it resolves *all* nested counters of the given name at the target, not just the innermost, and (b) it joins them with the provided separator string. This reuses the same target-resolution step (#2/#3 above) with the existing "all nested counters" logic from counters().

Error handling (Chromium interim behavior, pending spec clarification)

Since the spec does not yet define observable error behavior, this CL proposes:

- **No fragment / fragment doesn't resolve to an element / <url> is not a local reference:** the target-counter()/target-counters() value computes to the empty string (renders nothing), consistent with how attr() already degrades for missing attributes, and is **not** a parse error (the rest of content's value list still renders).
- **Target element found but named counter doesn't exist there:** treated the same as counter() referencing a counter that was never incremented. Resolves to the counter's initial/default value (typically as if it were 0 or 1, matching existing counter() behavior), not an error.

These behaviors will be called out explicitly in code comments and the CL description as "Chromium's interim interpretation of an underspecified area," and we'll file/track upstream spec issues if our resolution differs from what other engines (or a future spec revision) settle on.

Metrics

Success metrics

- New UseCounter entries for CSSValueID::kTargetCounter and kTargetCounters, to track real-world adoption (standard practice for new CSS features, consistent with go/newChromeFeature guidance).
- Correctness validated primarily via Web Platform Tests (see Testing plan) rather than a UMA-based "success" metric, since this is a long-tail authoring feature, not a perf or engagement feature.

Regression metrics

- No regression expected on the standard speed launch metrics, since this feature only activates when the new functional notations are present in author CSS. It adds no work to the hot path for existing pages.
- Watch Blink.UpdateLayout / paint timing on print-heavy benchmarks (if any exist in the perf trybot corpus) to confirm the layout-dependent resolution step (point 4 above) doesn't introduce unexpected extra layout passes for documents that do use the feature.

Experiments

Not Finch-gated. This is parser/rendering-correctness work for an opt-in CSS feature with no plausible default-behavior-change risk to existing content (no page today uses target-counter(), since it doesn't parse). Standard dev to beta to stable waterfall is appropriate.

Rollout plan

Waterfall. No external deadline; landing is quality-driven. Given the explicit scope split, the practical rollout sequence is:

1. Land target-counter() + target-counters() behind no flag (new CSS function, inert until used), once WPT coverage and review are complete.
2. File the target-text() follow-up design doc as a separate effort.

Core principle considerations

Speed

No impact on pages that don't use the new functions (inert addition to the value parser). For pages that do use it, the added cost is bounded by the size of the counter tree at the target node, comparable to existing counter()/counters() cost, plus one extra layout-dependent resolution pass analogous to existing string() page-content handling. No regression to the speed launch metrics is expected; will confirm via perf trybots before landing.

Simplicity

No new top-level UI or user-facing controls. This is a CSS authoring primitive, invisible to end users except as correctly-rendered cross-references in printed/PDF output. No new author mental model beyond what's already implied by the published spec examples. Authors who don't use the new functions see zero change. Print/PDF-focused tooling that currently relies on dedicated paged-media renderers (e.g. Prince, WeasyPrint, Vivliostyle) for this exact feature, as referenced in the bug thread's original report, would gain the option of using Chromium/headless Chrome instead.

Security

- **Restricting to local (same-document) fragment URLs only** (per spec guidance, see Spec status note) means this cannot be used to probe for the existence of cross-origin resources or leak information across origins. There is no new cross-origin attack surface.
- Target resolution reuses existing same-document ID lookup; no new untrusted-input parsing beyond what attr(href url) and existing fragment navigation already handle.
- No new IPC, no new privileged APIs, no renderer/browser trust boundary changes. This is pure renderer-side style/layout logic.

Privacy considerations

No new privacy considerations: no new data collection, no new fingerprinting surface (the feature only reads same-document counter state, which is already fully visible to any script in the page via the DOM), no cross-origin information disclosure given the same-document-only restriction above.

Testing plan

- New Web Platform Tests under css/css-content/ covering: basic target-counter() resolution, <counter-style> formatting, target-counters() nested-counter formatting and separator behavior, error cases (missing fragment, missing element, non-existent counter), and pagination-dependent cases (target on a different page than the reference, target counter value changing due to reflow).

- Reference/visual print tests (existing Chromium print-testing infrastructure) confirming correct page numbers render in --headless --print-to-pdf output for the canonical TOC and "see page N" cases from the spec.
- No special platform considerations expected. Paged media behavior should be platform-independent since it's driven by Blink's layout engine, not platform print dialogs.

Followup work

- File and land the target-text() design doc and implementation as the natural second phase.
- Revisit cross-document fragment references if there's future author demand and CSSWG resolves the relevant spec questions.
- Monitor the UseCounter data post-launch. If adoption reveals authors commonly hit the error cases in ways suggesting our interim error-handling choices are wrong, file an upstream CSSWG issue with real-world data.
- Clean up: none anticipated. No flags, no experimental-only code paths to remove.