PAPER · NOVEMBER 2026

Keep the Text: Measuring What Split-Text Effects Cost

A browser experiment comparing font shaping, document semantics and keyboard link stops. Browser-tree evidence does not replace screen-reader testing.

Slides Paper Data & methods Transcript Live comparison

Summary
Disclosure
Shaping is a font-dependent cost
Joined scripts make the change visible
A link can survive without its name
A label depends on the role
Our old tool removed the link
Inline is not unsplit
Keep the sentence intact when a highlight is enough
Moving pieces and the hidden copy
Objections answered
Recommendations
Open questions
Method
Sources

Summary

Before choosing a split-text animation, compare what it changes in the font's shaping, the document's semantics, and the keyboard's link stops. These costs depend on the library, split unit, CSS, font and text. Keep the sentence unsplit when a custom highlight is

enough. When pieces must move independently, inspect the actual page rather than treating an aria-label as an accessibility guarantee.

This is a browser experiment. Our measurements identify regressions; they do not replace testing with screen readers. The examples use pinned historical versions, not a census of the latest releases. Operate the comparison, view the slides, or inspect the dated data.

Disclosure

Overpunch makes speechType and other Type Tools. We've spent 15+ years building foundry websites, and Typetin is in development. That gives us an interest in this subject and an obligation to test our own implementation. The older speechType example below removes a link entirely. This paper recommends an intact-text platform feature before a wrapper-based effect when that feature meets the brief.

Shaping is a font-dependent cost

Kerning adjusts the spacing of pairs; ligatures and joining can shape several characters together. A span does not automatically stop that process. With matching inline formatting, shaping can continue across element boundaries. Independent inline-block boxes change the situation. CSS Text describes which boundaries must break shaping and which must not. Changing a piece's glyph-affecting style also matters. CSS Text 3, shaping across boundaries.

The October 7 experiment rendered the same five English lines in Merriweather and Source Serif 4, at 20 and 32 px. It compared intact text, plain inline character spans, individual inline-block characters, and intact text with kerning or kerning plus ligatures disabled. Spaces remained text nodes. These are controls for shaping, not performance measurements. Raw font results, measurement script.

OCTOBER 7 · CHARACTER BOXES · 32 PX

Merriweather: 546.75 → 562.48 px



+2.88%; inline spans: 546.98 px

Source Serif 4: 507.36 → 538.22 px



+6.08%; inline spans: 507.59 px

Same line: AVATAR Toward WAVE office affluent. Bars use a common 0–10% scale. Historical fontpick run, not a new benchmark. Raw measurements and method.

For AVATAR Toward WAVE office affluent at 32 px, Source Serif 4 measured 507.36 px intact, 507.59 px in inline spans and 538.22 px in character boxes: about 6.08% growth for the boxes. Merriweather measured 546.75, 546.98 and 562.48 px respectively: about 2.88%. These figures describe a deliberately demanding line. The ordinary sentence Yes, you can sign today. First read the official terms. grew from 713.98 to 727.44 px in Source Serif 4, about 1.89%. Neither percentage is a universal cost of splitting. October data and settings.

Intact text

AVATAR Toward WAVE office affluent

Inline spans, matching styles

AVATAR Toward WAVE office affluent

One inline-block per letter

AVATAR Toward WAVE office affluent

Live Source Serif 4 at 32 px, weight 400, default kerning and ligatures. Live generated spans illustrate shaping; they are not the archived measurement harness. The current WOFF2 and archived TTF builds have not been established as binary-equivalent, so these samples do not reproduce the archived widths. Matching inline formatting allows shaping across boundaries. CSS Text 3.

The live specimens above use the actual local font, with default kerning and ligatures. Their current rendering is illustrative, not a replacement measurement of the archived run. Disabling kerning on the original can reduce a width difference by removing some of what made the original well shaped; it does not restore that shaping in independent boxes. GSAP and SplitType document kerning-related shifts and mitigation. GSAP SplitText documentation, SplitType README.

Joined scripts make the change visible

A November 6 browser check selected the Arabic preset and GSAP Characters after waiting for fonts. The intact line measured 196.2 px; the split line 261.1 px, about 33.1% wider; matching inline spans measured 196.4 px. The phrase used the machine’s Arabic fallback font. This is evidence about one rendered phrase, not a result for all Arabic fonts or readers. November browser record.

Intact text

مرحبا بالعالم اليوم

Inline spans, matching styles

مرحبا بالعالم اليوم

One inline-block per letter

مرحبا بالعالم اليوم

Arabic specimen: intact, inline spans, then independent character boxes. Uses your system's Arabic serif fallback. November 6 browser check: 196.2 → 261.1 px with GSAP Characters (+33.1%); inline spans 196.4 px. The live boxes are generated spans, not the archived GSAP call. These archived 32 px values aren't a live measurement of this figure. Browser record.

Letters that should join are separated by independent boxes. The issue is larger than a decorative spacing preference. But the English and Arabic examples still need their font, size, browser and method attached. A number measured in a different font or before fonts load cannot stand in for these results.

A link can survive without its name

The comparison's ordinary sentence contains a real link. In GSAP 3.15.0 with type chars,words and default aria auto, the link remains in the document and focusable, but Chrome exposes an empty accessible name. Its descendant pieces are aria-hidden. Switching to words produces almost no width change in the historical short sentence, yet retains the name regression. The November browser checks independently observed the unnamed link in the live comparison. October library output, November observations.

The live page reports markup-derived roles, names, focusable links and text objects, plus rendered widths. These are browser-facing diagnostics. An empty name in Chrome is a specific regression we can identify without pretending we heard an announcement. It does not tell us which phrase a screen reader will speak in a particular navigation mode.

Intact sentence
1 link · name: the official terms

GSAP Characters or Words
1 link · empty accessible name in Chrome

speechType 1.0.11
Link removed

Wrapping in place
1 link · name: the official terms

November 6 local comparison of the default sentence. Link names checked against Chrome's accessibility tree; not a report of speech output. Evidence.

GSAP documents that its automatic labelling approach does not preserve nested semantics and offers a duplicate screen-reader-only copy as a workaround. That qualification belongs beside any account of the problem. The library also supports other ARIA configurations; this is not a claim that every configuration has the same failure. GSAP's accessibility guidance.

The historical tests also found two focusable links in anime.js 4.5.0's accessible configuration: an intact named link and an unnamed visual duplicate. That is an example of extra keyboard stops, not merely a theoretical risk. SplitType and Splitting with character-box CSS preserved a link but Chrome's computed name contained spaces between letters. Splitting without that box CSS retained the ordinary name. CSS and accessible-copy settings belong in a comparison, not just the package name. Pinned calls and Chrome focusable names.

A label depends on the role

WAI-ARIA 1.2 prohibits author naming on paragraph and generic roles. A labelled paragraph is not made conforming because one browser exposes its name. A heading is a different case: it can be named. The live comparison's element switch is consequential, not decorative. WAI-ARIA 1.2, roles which cannot be named, paragraph role, generic role.

Adrian Roselli reported screen-reader testing of GSAP's linked demo in issue 642, opened February 5, 2026. Two of eight screen-reader/browser pairings exposed the text. The tested host was a div. His later comment explicitly frames the issue as a request to improve the documentation's account of correct ARIA use, not a declaration that SplitText as a whole is broken. Those results apply to that demo and those combinations. They are external results, not our testing of this paragraph or current speechType. Issue and discussion.

We do not multiply that table into a general failure rate. It establishes why a browser name should not be treated as a guarantee of screen-reader output.

Our old tool removed the link

The October 7 published-package comparison ran speechType 1.0.11 and 1.1.0 on a paragraph containing a link. Version 1.0.11 reconstructed content from text, removing the link, and applied aria-hidden to six word spans. It also created a polite live region and wrote successive words into it. Version 1.1.0 retained the link and did not create those hidden-word attributes or a live region. Published-package output, script.

The old output file calls one field announced. That field logs DOM live-region updates driven by synthetic boundary events; it is not captured screen-reader speech. A live-region write can create competing announcements, but whether, when and how it is announced requires actual assistive-technology testing. We have not run that test.

The November live comparison reproduces the old version's removed link. Its ordinary inline word-wrapping alternative keeps the link name. This repairs identified regressions. It does not make wrapped words equivalent to intact text in every assistive technology. Browser evidence and limitations.

Inline is not unsplit

Ordinary inline spans can preserve shaping when adjacent pieces retain compatible formatting. They still add elements and can split text into more objects in the browser's tree. The November default character comparison reports three original text objects

versus 55 with inline wrapping; its word comparison reports 20. Object counts include exposed spaces, so counting only letter-bearing pieces is insufficient. These figures describe those settings, not a rule about reading pauses. Browser observations.

The active speechType word changes weight, the opsz axis and tracking; its CSS font size stays the same. A higher opsz value selects a design intended for a different viewing size, not an inherently larger glyph. In the measured specimen, the combined active styles change width and can cause a line to rewrap. Resting-width equivalence cannot demonstrate stable active layout. The top demo offers a manual word slider so this can be inspected without relying on speech-synthesis boundary support. Read-along demo.

A 2018 first-person report describes VoiceOver on iOS stopping at inline elements within a link. It is useful historical motivation to test fragmented text; it is not our current compatibility result and does not prove every inline span breaks reading. Daniel Göransson, Axess Lab.

Keep the sentence intact when a highlight is enough

The CSS Custom Highlight API associates ranges with a named highlight without wrapping the underlying text. It is an intact-DOM alternative when colour or background meets the brief. Highlights support a restricted set of painting properties, including text decoration and text shadow where supported, but not font weight, font size or transforms. It preserves the original DOM structure by construction; it does not certify screen-reader interaction or provide a semantic current-word announcement. CSS Custom Highlight API Level 1, highlight styling limits.

```
// range is a DOM Range covering the word to mark.
if ('highlights' in CSS && typeof Highlight !== 'undefined') {
  CSS.highlights.set('reading', new Highlight(range))
}
// To remove the visual mark:
// CSS.highlights.delete('reading')
```

```
::highlight(reading) {
  color: var(--background);
  background-color: var(--foreground);
}
```

Check support in the browsers you ship to and check the chosen colours. Released speechType 1.2.0 makes highlighting opt-in and retains spans as its default. It is additive, not a universal replacement for moving pieces. Version 1.2.0 is published on npm; the historical 1.1.0 measurements above remain dated comparisons, not new tests of 1.2.0. speechType 1.2.0 package.

Moving pieces and the hidden copy

The comparison also keeps intact text inside its original link, clips that real text, and places aria-hidden visual boxes beside it. The browser retains the name and the original text runs; the boxes can move. But the document contains duplicate text. Selection and find behavior need checking, and the relationship between visible content and clipped text may matter to touch exploration and focus highlighting. No screen reader has been run against this alternative. A tidy browser tree is insufficient to recommend it without those checks. Operate the boxes-beside-real-text option.

This alternative retains the boxes' shaping costs. It is not an argument that every motion effect requires splitting: whole-element transforms and other designs may avoid per-piece markup entirely.

Objections answered

THE WIDTH CHANGE IS SMALL

It can be. The ordinary English line and word-level split are useful counterexamples to a dramatic universal story. The named-link regression can remain even when width barely

changes, and joining scripts can show a much larger change. Pick representative text, not only a striking specimen.

THE PARAGRAPH HAS A LABEL IN CHROME

That is an observation about Chrome, not permission to name a prohibited role and not a measurement of spoken output. Use the role appropriate to the content and verify the nested link independently. The external div test is a warning against substituting a label for retained semantics.

DIMMING MAKES THE ACTIVE WORD EASIER TO SEE

It may help distinguish it visually, but it also lowers inactive-text contrast. The shipped 1.2.0 default is 0.6, an accessibility-motivated change; it can still fail AA on particular foreground/background colours. Its effect depends on compositing and the chosen colours, not a single opacity threshold. We do not claim blanket AA or a reader benefit from that default. WCAG contrast criterion.

Recommendations

1. Keep an intact-text baseline with its real links. Record the font, styles, language and actual library call.
2. Prefer a platform highlight if colour/background meets the brief. Prefer whole-element motion if independent pieces are unnecessary.
3. If wrapping is needed, retain existing semantics and inspect shaping with the final CSS, including the active effect, not just the resting state.
4. Check keyboard focus stops and their names against the original. Inspect the browser's accessibility tree independently of the page's own diagnostics.
5. Check contrast on the rendered colours, zoom and reflow, selection, find, focus highlighting and reduced motion.
6. Test remaining questions with real screen readers in the combinations your audience uses. Name those combinations when reporting results.

Open questions

We have not measured comprehension, reader preference, performance, speech-synthesis timing or screen-reader announcements. We do not know whether each screen reader will join fragmented text smoothly, how moving aria-current is handled, or whether the clipped-copy alternative behaves well under touch exploration. Those are follow-up tests, not conclusions from browser-tree object counts.

Method

There are separate evidence sets. October 7, 2026: archived Chromium 149.0.7827.55 / Playwright 1.61.0 runs of pinned published packages and controlled font specimens. November 6, 2026: bounded checks of the live local comparison, with direct Chrome accessibility-tree captures and a settled-font Arabic measurement. These are not merged into a trend, a compatibility census or a performance result. Methods, raw files and reconciliation.

The library harness includes an error in its GSAP call without explicit split options; we do not count that failed case as a library accessibility result. Successful chars,words and words calls provide the GSAP examples here. Motion+ 3.0.0 appears only in the archived harness, not in the live comparison. Splitting's no-CSS and box-CSS cases are distinct. The font control uses plain generated spans rather than a library, so it isolates formatting boundaries; it does not benchmark speechType itself.

The larger November browser sweep is not claimed clean. Its initial 1,603 states / 4,809 column readings reported 198 accessible-name mismatches involving non-breaking spaces: 90 plain, 90 kept, 18 split. After preserving non-breaking spaces in the diagnostic, a November 7 rebuilt-site reproduction checked 91 states / 273 column readings and found six remaining host-name spacing mismatches, all in anime.js heading cases. This bounded run does not certify the original sweep. The page's computed readings remain an approximation, not a substitute for direct browser-tree inspection. This paper's specific link examples rely on bounded direct captures, and its widths rely on rendered measurements. Data status.

Sources

- WAI-ARIA 1.2: prohibited naming and role definitions. Normative basis for the paragraph/generic distinction; checked November 6.
- Accessible Name and Description Computation 1.2. Algorithmic context for hidden descendants and names; browser-tree checks remain separate.
- CSS Text 3: shaping across element boundaries. Basis for distinguishing compatible inline spans from independent character boxes.
- GSAP SplitText documentation. ARIA options, nested semantics, duplicate-copy workaround and kerning guidance. Historical package tested: 3.15.0.
- Adrian Roselli: GSAP issue 642 and comments. External div test; two of eight combinations, with documentation scope explicitly clarified. Checked November 6.
- CSS Custom Highlight API Level 1 and CSS Pseudo-Elements: highlight styling. Range-based highlighting and layout restrictions. API source checked November 6.
- SplitType source and README, Splitting source, anime.js source. Context for pinned calls; the archived package output, not current main branches, supplies our results.
- Daniel Göransson: Text Splitting Causes Screen Reader Problems. First-person 2018 VoiceOver/iOS report; not a current retest.
- WCAG 2.2: Contrast (Minimum). Contrast requirements do not imply one safe opacity for all colours.
- Our measurements and reproducibility notes. Separate October and November records, scripts and known errors. No simulated reviewer opinions are used as attendee evidence.