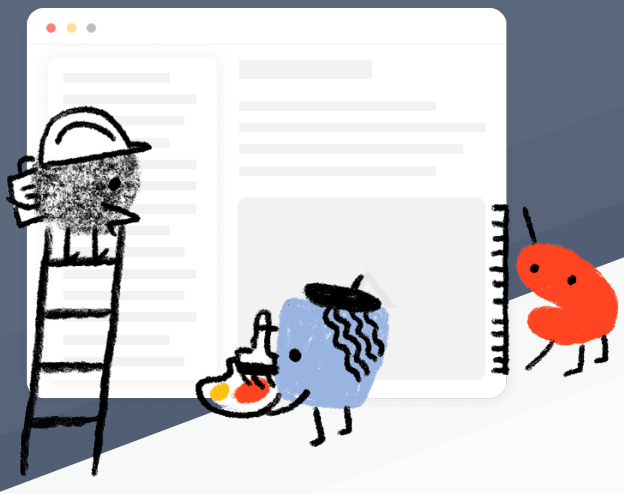
2025 Web Almanac

HTTP Archiveの年次報告書
ウェブの状態レポート



目次

導入

部 I. ページコンテンツ

部 II. ユーザー体験

部 III. コンテンツ公開

部 IV. コンテンツ配信

付属資料

序章

ウェブは今日、私たちの現代デジタルインフラに不可欠な存在となっています。ウェブアプリケーションは、モバイルアプリから衛星通信で使われるエンドポイントまで、日常生活に関わるあらゆる技術環境で見られるようになりました。Web Almanacでは、ウェブのトレンドと進化を理解することを目指しています。ウェブを愛する者として、ウェブに関する主要な技術レポートの1つである2025 Web Almanacの第6版をお届けできることを嬉しく思います。

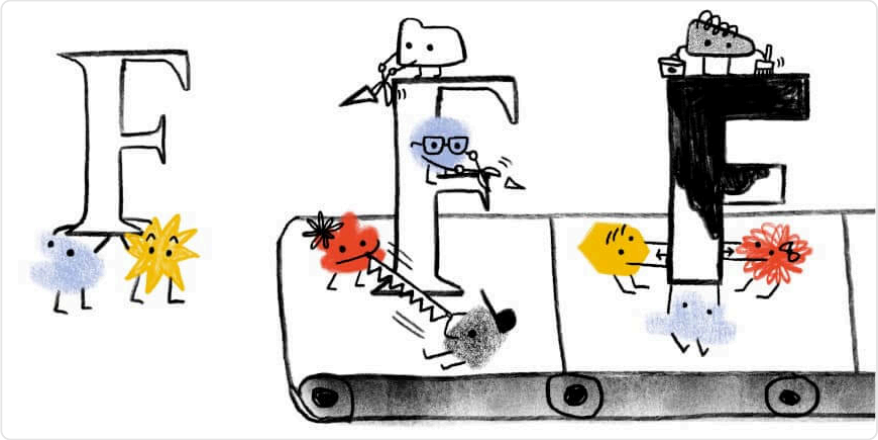
Web Almanacはオープンソースプロジェクトであり、世界中の専門家（研究者、開発者、コンサルタント、編集者）によって貢献されています。今回もコミュニティの素晴らしい貢献を体験できた、素晴らしい版となりました。このレポートを可能にしたのは、コミュニティの皆さんの積極的な関与があってこそです。

このコミュニティ主導のプロジェクトは1720万のウェブサイトを分析し、HTTP Archiveが提供する244TBのオープンソースデータを処理しています。本レポートでは、アクセシビリティ、パフォーマンス、セキュリティなどさまざまな側面を分析し、ウェブがどのように進化しているかを理解し、構造的・技術的・社会的な発展に関する洞察を深めます。

このレポートがコミュニティのウェブの現状理解に役立つことを願っており、この広範な成果をぜひ探求していただければと思います。

—Nurullah Demir, 2025 Web Almanac 編集長

部 I 章 1
フォント



Charles Berret によって書かれた。
Bram Stein と José Solé によってレビュ。
Charles Berret と Ivan Ukhov による分析。
Chris Lilley 編集。
Sakae Kotaro によって翻訳された。

はじめに

HTTP Archiveが2011年にウェブタイポグラフィのデータ収集を開始した当初、カスタムウェブフォントの使用率は一桁台前半に過ぎませんでした。当時、`@font-face`を使ってセルフホストのフォントを配信したり、Typekit、Fonts.com、Google Web Fontsなどのサービスを利用したりしていたウェブサイトは、わずか約3〜5%でした。2011年当時の残りのサイトはすべて、ユーザーのデバイスで利用可能な少数の「ウェブセーフ」なシステムフォント（Arial、Courier、Timesなど）のみを使用していました。

デザイナーたちがカスタムタイポグラフィによってビジュアルアイデンティティを他のサイトと差別化できると気づくと、ウェブフォントは急速に標準となっていきました。2015年までにウェブフォントはウェブサイトの半数以上で使用されるようになり、2020年には約75%の普及率に達しました。現在、問われるのはウェブサイトがウェブフォントを使用するかどうかではなく、どの書体を表示し、フォント機能を使って表現の可能性を最大限に引き出しているかどうかです。

同時に、フォントの配信方法も変化しています。サードパーティのCDNにのみ依存するのではなく、フォントファイルをセルフホストすることを選ぶサイトが増えており、多くのサイトが両方のアプローチを組み合わせて使用しています。昨年のデータでは、セルフホスト専用の利用が明確に増加し、セルフホストと外部サービスを組み合わせるサイトが減少傾向にあることが示されており、このトレンドが2025年も続いているかどうかを調査します。

この章では、現在のウェブタイポグラフィのいくつかの側面を探ります。どのプロバイダーが最大のシェアを持つか、フォントがどのように読み込まれ速度のために最適化されるか、実際のCSS実装でどの書体がリードしているか、非ラテン文字スクリプトがどのように扱われるか、そして可変フォントとカラーフォントがウェブタイポグラフィの未来にとって何を意味するかを考察します。

ウェブフォントの使用状況

ウェブフォントの使用は増加を続けていますが、ウェブ全体（最大のCJK文字セットを除く）でウェブフォントの普及が飽和に近づくにつれ、その成長は自然と鈍化しています。

図1.1. ウェブフォントを使用しているモバイルページの割合。

2025年には、ウェブフォントは約88%のウェブサイトで見られ、2024年の約87%からわずかに増加しました。（ここでの指標は、ウェブフォントファイルをリクエストするページのシェアであり、実質的にウェブフォントを使用しているウェブサイトの割合を表しています。）

この広範な使用状況は、開発者がコアシステムフォントのみに限定されていた時代からウェブタイポグラフィがいかに進化したかを示しています。ただし、注目すべき少数派（約12%のサイト）はいまだに古いデフォルトフォントを使い続けています。

ホスティングとサービス

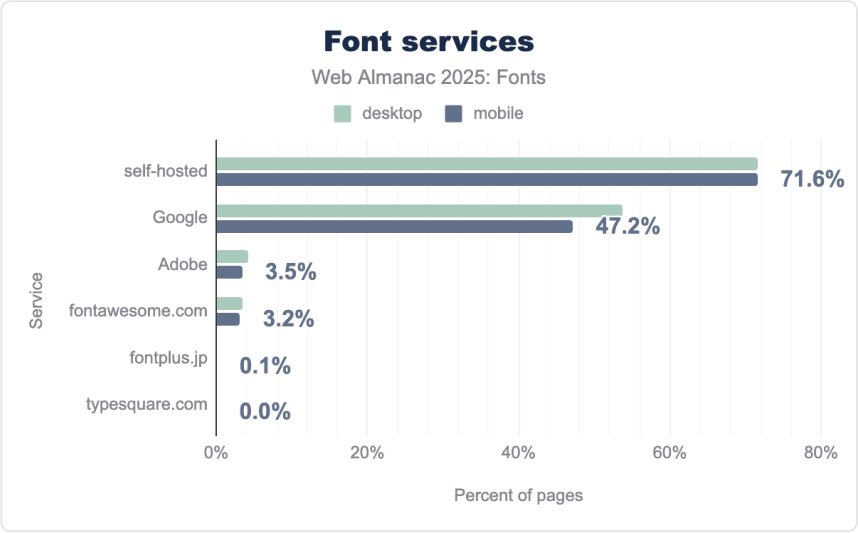


図1.2. フォントサービス。

今年のWeb Almanacクロールにより、セルフホスティングがウェブフォント配信において最も広く普及した手段であることが確認されました。約72%のウェブサイト（デスクトップ・モバイルともに）が、セルフホスティング専用または自己ホストと外部ホストの組み合わせで、少なくとも1つのフォントファイルを自社のオリジンから提供しています。この数字は昨年より1ポイント強高く、2022年比では約4〜5ポイント高く、着実な増加を示しています。つまり、2022年に顕著になったセルフホスティングの増加は今なお続いています。

この72%の内訳として、2つの重要なグループが挙げられます。

- **セルフホスト専用フォント**: デスクトップサイトの約31.5%、モバイルサイトの約36.4%が、セルフホストフォントのみを使用しています（サードパーティのフォントサービスは一切使用なし）。これらを平均すると、2025年には全サイトの約3分の1がセルフホスティングのみに依存しています（昨年の約30%から増加）。これは大きな変化であり、クリティカルパスから外部プロバイダーを排除し、すべてのフォントを自社ホストする決断をしたサイトが増えています。

- **ミックスホスティング（セルフホスティング＋サービス）**: デスクトップサイトの36.1%、モバイルサイトの31.7%が、同一ページでセルフホストフォントとフ

> ォントサービスを組み合わせて使用しています。これらのほとんどの場合、サービスはGoogle Fontsで、アイコンCDNや別のソースが追加されることもあります。この「ハイブリッド」アプローチは昨年（2024年のデスクトップ約38.5%、モバイル約32.7%）よりやや一般的でしたが、一部のサイトがセルフホスティングへ完全移行したため、若干減少しています。

これらのグループを合わせると、約72%のサイトが何らかの形でセルフホストしています。残りの約28%のサイトは、フォントをサードパーティサービスに完全依存しており（そのカテゴリでは断然Google Fontsが最も選ばれています）。

ウェブフォントサービス

セルフホスティングの成長にもかかわらず、ホスト型フォントCDNはGoogleの圧倒的な存在感により、依然として全ウェブサイトの約半数に表示されています。今年のクロールでは、Google FontsはデスクトップサイトのはGoogleのAPIを使ってフォントを取得しているサイトすべてをカウントしており、Googleが唯一のソースであるか他のソースと一緒に使用されているかを問いません。）デスクトップにおけるGoogleのリーチは、2024年の57%から約3パーセントポイント低下し、2022年比では約6ポイント低下しており、セルフホスティングへのシフトを反映しています。モバイルでは、Googleのシェアはわずかにしか低下しておらず（昨年比約0.3ポイント、2022年比約4ポイント）、全体的なパターンは2024年に指摘されたものと同じです。Google Fontsは依然として多くのサイトでデフォルトの選択肢ですが、より多くのサイトがセルフホストするにつれて、その優位性は徐々に低下しています。

他のフォントホスティングサービスは比較すると非常に小規模です。Adobe Fonts（旧Typekit）は再び緩やかな成長を見せました。2025年には、デスクトップサイトの約4.2%、モバイルサイトの約3.5%がAdobe Fontsを使用しており（合計で約3.8%）、昨年の約4.1%から増加しています。これはウェブ全体では小さなシェアですが、実質的な成長（2年間で10〜11%の相対的増加）を示しています。

人気のアイコンフォントCDNであるFont Awesomeは緩やかな減少傾向にあります。2022年には約4.4%のサイトで使用され、2024年には4.0%、2025年には約3〜4%のサイトに減少しています。この減少は、多くのサイトがインラインSVG、アイコンスプライトシート、セルフホストアイコンセットなど代替のアイコンソリューションに移行していること、またパフォーマンス上の理由から大きなアイコンフォントの読み込みを避ける開発者がいることが原因と考えられます。それでも、約25サイトに1サイトでFont Awesomeが継続的に使用されていることは重要であり、それに匹敵するリーチを持つ単一のフォントファミリー（アイコンまたはテキスト）はほとんどありません。

その他のサービスは、ウェブ規模ではほぼ無視できる程度です。MonotypeのFonts.com、

Extensis、Cloud.Typography、Type Networkなどのレガシーサービスはそれぞれ1%未満のサイトにしか対応していません。これらの多くはクロールデータでほぼ見えなくなるほど減少しています。実際問題として、2025年にウェブサイトがホスト型フォントサービスを使用している場合、ほぼ確実にGoogle Fontsです。Adobe FontsやFont Awesomeである可能性は低く、それ以外である可能性はさらに低いです。

フォントホスティングの組み合わせ

ほとんどのサイトがウェブタイポグラフィに1つのソースだけを使用していないため、どのように配信を組み合わせているかを見ることは有益です。実際にはウェブはいくつかの予測可能な組み合わせ（セルフホスティングのみ、Googleのみ、またはその2つのハイブリッド）に収束しています。

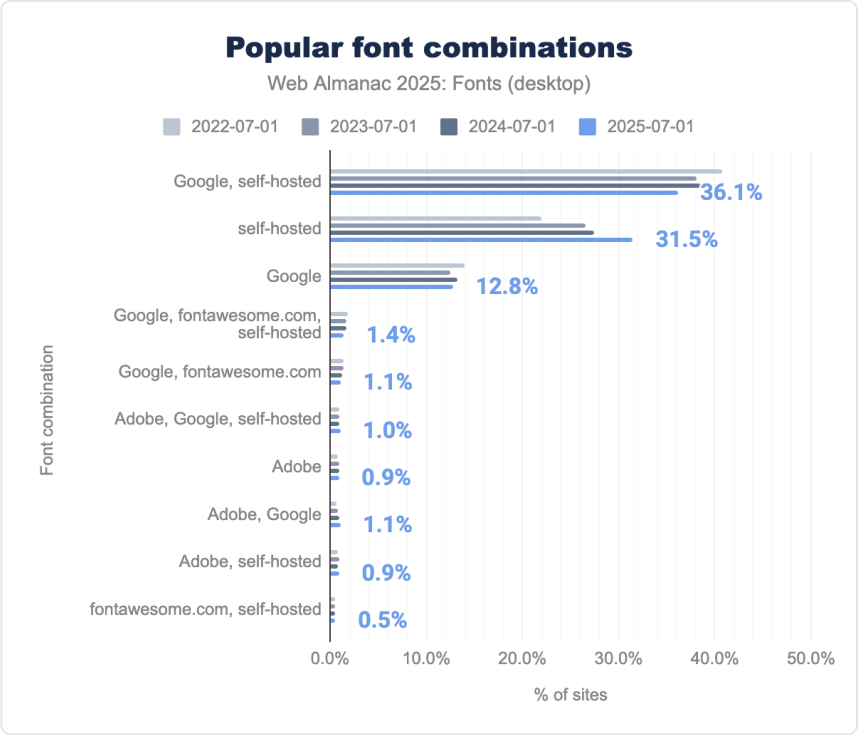


図1.3. デスクトップサイトで人気のフォントホスティングの組み合わせ。

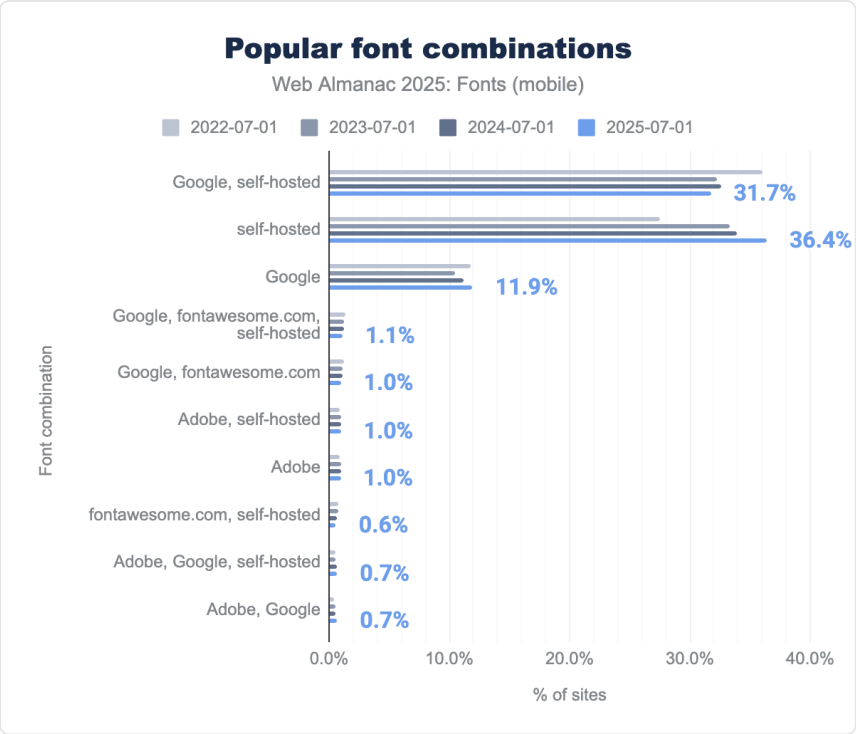


図1.4. モバイルサイトで人気のフォントホスティングの組み合わせ。

1. **Google Fontsとセルフホストフォントの組み合わせ**: デスクトップサイトの約36%、モバイルサイトの約32%が使用。このハイブリッドアプローチは非常に一般的です。例えば、サイトがブランドやアイコンフォントをセルフホストしながら、本文テキスト用にGoogle ホストのフォントを使用する（またはその逆）といったケースです。

2. **セルフホストフォントのみ**: デスクトップサイトの約32%、モバイルサイトの約36%が使用（全体では約34%）。ページ上のすべてのカスタムフォントがサイト自身のドメインから提供されることを意味します。前述のように、このシェアは昨年の約30%から今年は全サイトの約3分の1まで成長しています。

3. **Google Fontsのみ**: デスクトップサイトの約13%、モバイルサイトの約12%が使用。これらのサイトはフォントのニーズをGoogleのCDNに独占的に依存しています（多くの場合、Googleのスタイルシートへの Standardの `<link>` を通じて）。

これら3つの構成が合わさって、全ウェブサイトの約5分の4を占めています。その他の組み

合わせ（GoogleとAdobeの併用、Adobeとセルフホスティングなど）は、クロールデータではすぐに希少になります。実際、上位のパターンを超えると、特定の組み合わせのシェアは1%を下回り、8番目や10番目に一般的な構成になると、その使用状況はほぼ知覚不可能（サイトの約0.05%）になります。

これらの一般的なサービスの組み合わせを超えて、年々バランスはセルフホスティングへと傾いています。2024年には約30%のサイトが専用セルフホストでしたが、現在は約34%です。対応して、「ミックス」アプローチ（セルフホスティング＋サービス）は若干縮小しています（例えば、デスクトップでは約38%から約36%）。つまり、両方を行うことで保険をかけるサイトは減少し、すべてを自社ホストすることを決定したサイトが増えています。これはパフォーマンスとプライバシーの動機によるものと考えられます。昨年の章で指摘されたように、キャッシュパーティショニングなど現代のブラウザの変更が、Google CDNを使用することの古いパフォーマンス上の優位性を取り除きました。

フォントファイルのホスティング

フォントホスティングを見る別の方法として、どの特定のフォントファイルが最もリクエストされているか、またそれらがセルフホスティングからかサービスからかを確認することが挙げられます。これにより、ウェブで最も人気のあるフォントファミリーとその配信方法についての洞察が得られます。

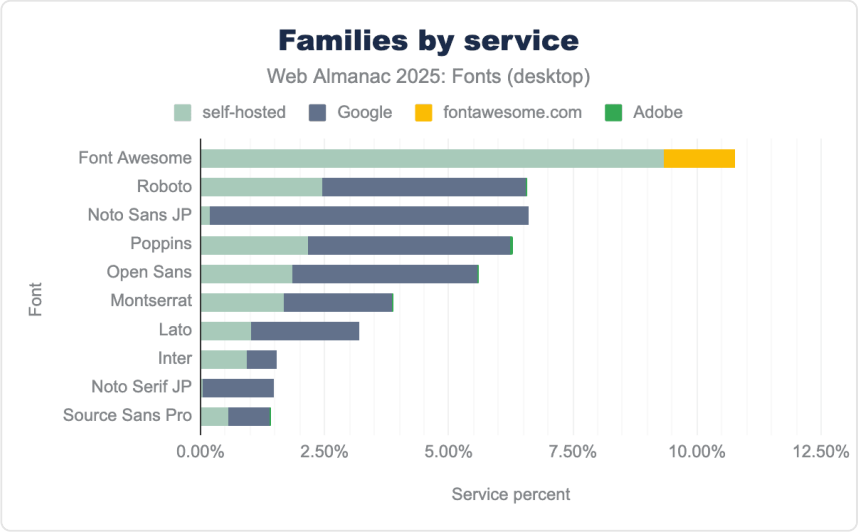


図1.5. サービス別ファミリー（デスクトップ）。

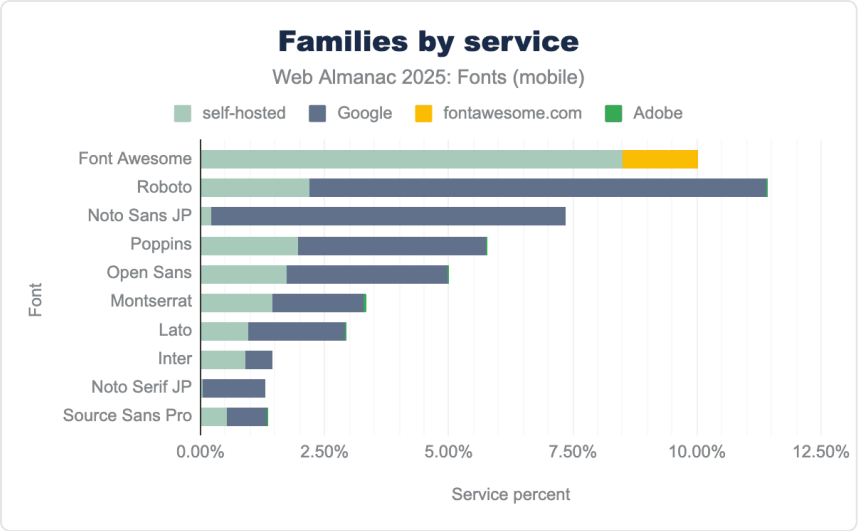


図1.6. サービス別ファミリー（モバイル）。

セルフホスト側では、トラフィックはアイコンフォントファイルと一般的なライブラリアセットに集中しています。Font Awesome（アイコンセットのTTF/WOFFファイル）のローカルコピーは、2025年のクロールではデスクトップのフォントリクエストの約8.5%、モバイルのフォントリクエストの約9.3%を占めています。これにより、Font Awesomeはテキスト書体ではなくアイコンセットであるにもかかわらず、ウェブで最も頻繁にフェッチされるフォントリソースの1つとなっています。注目すべきことに、Font Awesomeの使用のほとんどはセルフホスティング経由であり、`fontawesome.com` CDNから直接フェッチされる割合はずっと少ないです。その他の人気のユーティリティフォントも同様のパターンを示しています。例えば、IcoMoon（別のアイコンフォントジェネレーターの出力）はデスクトップの約1.0%、モバイルのフォントリクエストの1.1%を占め、ETmodules（WordPressテーマでよく見られる）はデスクトップで約0.5%、モバイルで0.7%です。これらもほぼ常に、それらを使用するサイトに常駐するセルフホストフォントです。実際には、ウェブ上でのこの「セルフホスティング」のセグメントは、オーダーメイドのテキストフォントのセルフホスティングだけでなく、これらの一般的なアイコン/フォントアセットを自サイトのファイルにバンドルすることについてです。

Google Fonts側では、フォントリクエストは典型的なロングテール分布を示しています。先頭にある少数の非常に人気のあるフォントと、使用率が低い多くのフォントです。デスクトップでは、Robotoが単一で最もリクエストされるGoogle ホストのフォントであり、すべてのGoogle フォントリクエストの約9.2%を占めています。その直後にNoto Sans JP（日本語）があり、Googleのリクエストの約7.1%を占めています。これはGoogleのリーチがいかにグローバルであるかを即座に示しています。上位2つのフォントのうち1つはラテン/英語のUIフェイスで、もう1つは日本語フォントです。その下のリストは2025年に予想される通りで

す。Poppins（デスクトップでのGoogle フォントリクエストの約3.8%）、Open Sans（デスクトップでGoogle経由で約3.3%）、Lato（約2.0%）、Montserrat（約1.9%）。モバイルでは順序がわずかに変わります。Noto Sans JPが実際には約6.4%で1位であり、Robotoは4.1%ですが、Poppins（約4.1%）、Open Sans（約3.7%）、Lato（約2%）、Montserrat（約2.2%）はすべて同じトップグループにいます。要するに、Google Fontsの最大量フォントには、広く使用されているラテン系ファミリーと重要な非ラテン系フォント（特にCJKスクリプト）の両方が含まれています。

これらのフォントのうち、どれがセルフホストされる傾向があり、どれがGoogleから提供される傾向があるかを注目するのは興味深いことです。例えば、Noto Sans JP（大きな日本語フォント）は、データではほぼ完全にGoogleのCDNによって提供されています。Noto Sans JPのセルフホスト率はリクエストのわずか約0.2%であり、そのフォントを使用するほぼすべてのユーザーがGoogleに配信を任せることを選んでいることを意味します（おそらく非常に大きなファイルで、Googleのインフラがそのために最適化されているためです）。Noto Serif JPも同様で、リクエストの約1.3〜1.4%に表示されます（ほとんどがGoogle ホスト）。対照的に、多くの人気のラテン文字系ファミリーはGoogleによって提供される使用量に加えて、セルフホスト分もかなり存在します。Robotoもセルフホストされています（デスクトップのフォントリクエストの約2.2%がセルフホストのRobotoファイル用で、Googleからの9.2%に加えて）。同様に、Poppinsは約2.0%がセルフホスト、Open Sanは約1.7%、Montserratは約1.4%、Latoは約1.0%、Interは約0.9%がセルフホスト（デスクトップ数値）。これらの各ケースでは、同じフォントが多くのサイトでGoogleによっても配信されています。最も人気のあるGoogle Fonts（Roboto、Open Sans、Poppins、Montserrat、Lato、Inter）がセルフホストデータでも無視できない率で表示されること、そしてセルフホスティングが成長しGoogleのシェアが緩和されていることから、多くのプロジェクトがGoogle Fontsから始めてパフォーマンスやプライバシー上の理由から同じファミリーを社内に取り込む可能性が高いです。（注：データは重複と方向性のシフトを示せますが、サイトごとの移行パスは示せません。）

Adobe Fontsは、キットが必要で多様な独自の商用フォントを提供する別種のサービスであるため、異なるプロファイルを持っています。単一のAdobe提供フォントがシェアを独占していません。2025年のトップAdobe フォントファミリーはProxima Novaで、デスクトップとモバイル両方のフォントリクエストの約0.8〜0.9%を占めています。その下では、Adobeのトラフィックは長いファミリーリストに分散しています。これらは主にFutura PT、Brandon Grotesque、Freight Sans、Acumin、Sofia Pro、Aktiv Grotesk、Museoなどの人気のブランディングおよび編集向けフェイスで、それぞれリクエストの数百分の一パーセントから半パーセント程度を占めています。これはAdobe Fontsが一般的にどのように位置づけられているかと一致しています。何百万ものサイトで1つのフォントが見つかるというよりも、それぞれ比較的少数のサイトで使用される非常に優れたフォントを幅広く提供しています。これは本質的にGoogleのパターンとは逆であり、Googleでは少数のフォントが使用量の大部分を占めています。

パフォーマンス

2025年のウェブにおけるタイポグラフィのパフォーマンスは、ウェブフォントをできるだけ小さく、クリーンな形で配信することに重点が置かれています。このセクションでは、最も重要な2つのパフォーマンスの要素を見ていきます。フォーマット（ウェブは基本的にWOFF2に集約し、WOFFがフォールバックとして残っている）と、それらのフォーマットが生み出す実際の転送サイズ（中央値は十分に合理的ですが、ロングテールは本当に遅くなる可能性がある）です。これらの結果をウェブフォントの衛生チェックリストとして読む場合は、WOFF2を使用し、MIMEタイプを確認し、必要に応じて大きなフォントをサブセット化し、セルフホストしている場合は特に注意してください。そこに非効率性のほとんどがまだ存在しています。

ファイル形式と最適化

ウェブでのフォントファイル形式に関しては、モダンなWOFF2フォーマットが引き続きリードしています。WOFF2（Web Open Font Format 2）は優れた圧縮を提供し、より速い転送のためにファイルサイズが小さくなり、すでに何年もすべての主要なブラウザでサポートされています。2025年には、WOFF2はデスクトップとモバイルページの両方でフォントファイルリクエストの約65%を占めています。これは現代のフォントフォーマットの圧倒的多数であり、ウェブの標準フォントフォーマットとしてのWOFF2への継続的な集約を表しています。言い換えれば、今年のデータのすべてのウェブフォントの約3分の2がWOFF2フォーマットです。WOFF2への集約は、より優れた圧縮がより速いフォントの読み込みとより少ない帯域幅の使用につながるため、パフォーマンスにとっての良い傾向です。

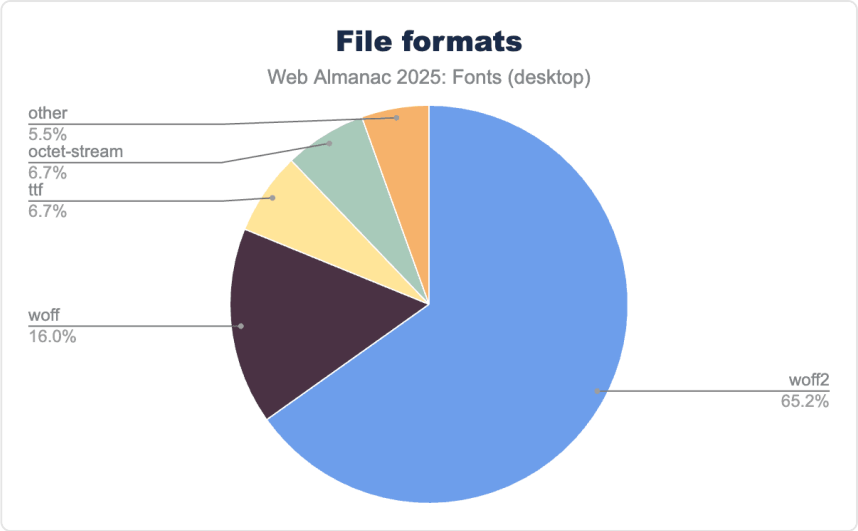


図1.7. デスクトップサイトのフォントファイル形式。

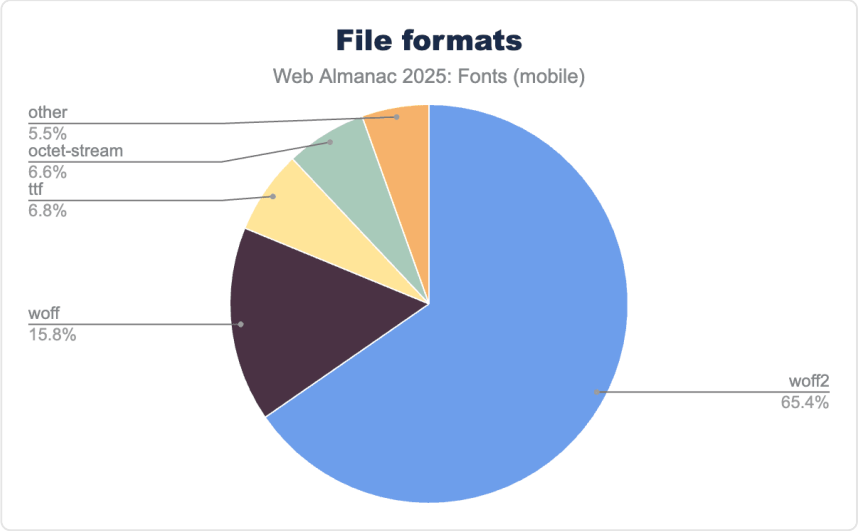


図1.8. モバイルサイトのフォントファイル形式。

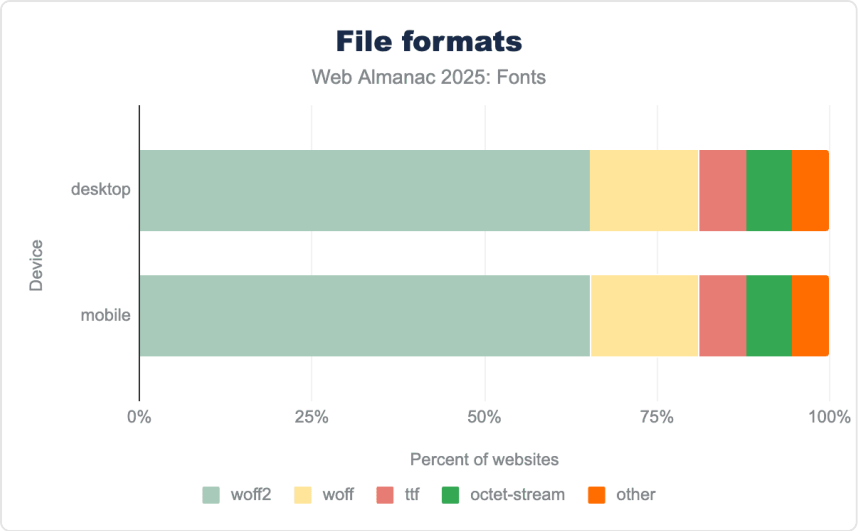


図1.9. デバイス別のフォントファイル形式。

WOFF2のすぐ後ろには古いWOFF 1.0フォーマットがあり、依然として重要なフォールバックです。2025年には、WOFF 1.0はデスクトップとモバイルの両方でフォントリクエストの約16%を占めています。WOFF2の65%のシェアと合わせると、2つのWOFFフォーマットが合計でフォントリクエスト全体の約81%を占めています。これはWOFF2を主要フォーマットとした継続的な集約を反映しており、WOFF 1.0は主にレガシーおよび互換性のケースのために残っています。WOFF2の広範なブラウザサポートによってほぼすべてのユースケースがカバーされているため、より多くのサイトやフォントプロバイダーがWOFFからWOFF2に移行しています。WOFFは2010年代初頭に導入された当時は大きな改善でしたが、現在では主に古いブラウザや長期間フォントファイルを更新していないサイトで使用される、主にレガシーのフォールバックとなっています。

WOFFとWOFF2を超えると、ほとんどがエッジケースとして残るのはマイナーなフォーマットだけです。以下で述べるように、来年のIncremental Font Transfer技術の登場は大きなフォントにとって特に価値のある改善となり、WOFF2のシェアを削り始める可能性があります。それでも、注目すべき割合のフォントが依然として生のTrueType（`.ttf`）ファイルとして提供されています。2025年には、フォントリクエストの約6.7%がTTFであり、多くの場合、元のフォントファイルをWOFFとして再パッケージせずに提供するCDNやセルフホスティングのセットアップによるものです。同様に、フォントリクエストの約6.6%が汎用の`application/octet-stream` MIMEタイプで配信されています。`octet-stream`は独自のフォントフォーマットではないことに注意してください。これは通常、サーバーが適切なフォントMIMEを指定していない設定ミスを示します（例えば、WOFF2ファイルが正しい`font/woff2`の代わりに`Content-Type: application/octet-stream`で提供されるかもしれません）。2025年にも、この設定ミスのMIME問題が残念ながら続いており、フォン

トファイルの約6〜7%が依然として誤ったMIMEタイプで送信されています。これは主に、設定の問題を持つ少数の主要ホスト（昨年の分析ではcdnjsとWixのCDNが顕著な問題サイトでした）によるものです。`font/woff2` や `font/woff` などの正しいMIMEタイプを使用することでコンテンツの処理とデバッグが改善されるため、これは小さいながらも注目すべきスライスです。

重要なのは、モダンなフォーマットを組み合わせると、WOFF2 + WOFFがウェブフォントエコシステムの5分の4以上を占めることです。OpenType/CFF（`.otf`）やSVGフォントなどの他のフォーマットはデスクトップでは健在ですが、公共ウェブからは事実上消えており、EOT（旧IE固有のフォーマット）はほぼ絶滅しています。2020年に、Web Almanacはすでに SVGとEOTがほぼなくなったと指摘しており、その傾向は続いています。典型的なサイトのクロールで非WOFFフォントファイルに遭遇することは今や稀です。圧倒的なアドバイスは変わりません。ウェブフォントにはWOFF2を使用し、特定のニーズがない限り古いフォーマットは廃止してください。今年のクロールデータは、ほぼすべての人がWOFF2に収束し、WOFFをトレーリングフォールバックとして使用し、その他はすべて無視できる程度であることを強く裏付けています。

ファイルサイズとパフォーマンス

パフォーマンスの観点から、モダンなフォーマットを使用することは重要です。なぜならフォントファイル自体が平均的に大きくなってきているからです。昨年の分析では、ウェブフォントファイルの中央値サイズが以前の年と比べてかなり増加しており、サイトがより複雑なフォント（例えば、複数の言語をサポートするためにより多くのグリフを持つフォントや、多くのスタイルをバンドルする可変フォント）を使用していることを示唆していました。2025年のデータは、このより大きなフォントファイルへの傾向が続いており、最適化がさらに重要であることを示しています。

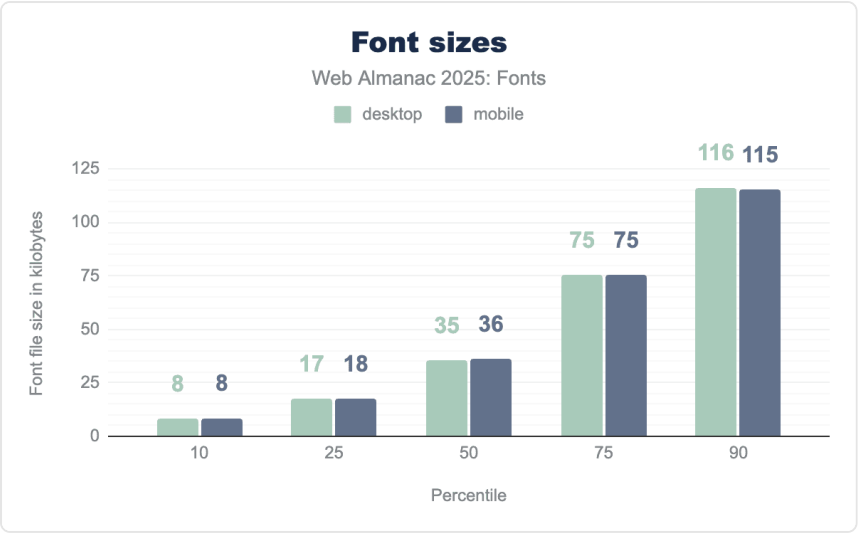


図1.10. フォントファイルサイズ。

ウェブでのフォント転送サイズの分布を見ると：

- 2024年には、フォントファイルの中央値サイズは約36〜39 KB（gzip圧縮）で、2年前の30 KB台前半から増加しています。2025年には、中央値は引き続き30 KB台後半から40 KB台前半の範囲（デスクトップでは30 KB台中盤、モバイルではわずかに高い）にあり、典型的なフォントファイルが過去1年でそれほど成長していないことを示しています。より大きなサイズでの横ばいとなっています。

- 75パーセンタイルでは、2024年のフォントファイルは約76〜77 KBでした。2025年では、これは概ね同様（70 KB台中盤）です。したがって、ウェブフォントの4分の3が80 KB未満であり、特にWOFF2圧縮でも依然として管理可能です。

- ただし、90パーセンタイルでは、2024年にはフォントファイルが約112〜116 KBでした。2025年のクロールでは、90パーセンタイルが115〜116 KBの範囲で同様です。したがって、フォントのトップ10%はかなり大きく（100 KBをはるかに超えています）。これらは複雑なスクリプト用のフォント（何千ものグリフを持つCJKフォント）、多くの軸を持つフル機能の可変フォント、または単にサブセット化されていないファイルである傾向があります。

- 分布の末尾は非常に重くなっています。99パーセンタイルでは、2024年のデスクトップのフォントファイルは約776 KB（モバイルでは572 KB）でした。2025年では、最大の1%のフォントは依然として数百キロバイトの範囲にあります。これらはおそらく大規模なCJKフォント、埋め込みグラフィックを持つアイコンフォント、またはデバッグテーブルが誤って含まれたフォントです。ほとんどのフォントは合理的なサイズですが、少数の外れ値が非常に重くなる可能性があることを思い起こさせます。

ウェブサイトのほとんどは適度なサイズのフォントファイルを提供していますが、非常に大きなフォントファイルも無視できない数存在するというのが主な学びです。これにより、効率的なフォーマットとテクニックを使用することが重要になります。WOFF2はファイルサイズを圧縮することで大いに役立ちますが、フォントが非圧縮で半メガバイトであれば、WOFF2でも限界があります。サブセット化（つまり、必要なグリフのみを含む）などのテクニックは、CJKやアイコンフォントなどの非常に大きなフォントにとって不可欠です。

WOFF2の普及が大きなフォントファイルのパフォーマンスへの影響を軽減するのに役立つのは心強いことですが、フォントをセルフホストするサイトは平均的に最適化の余地が多いことも判明しました。例えば、2024年にはセルフホストのフォントリクエストの約58%のみがWOFF2でした（全体では約78〜81%）。セルフホストのセットアップはWOFF（約18〜19%）やもいくつかの生のTTF（約6%）のシェアが大きかったです。これは、フォントをセルフホストする一部の開発者が古いファイルを使用しているか、WOFF2に圧縮していない可能性があることを示唆しており、一方でGoogleのような大手プロバイダーはブラウザがサポートしていれば常にWOFF2を提供します。

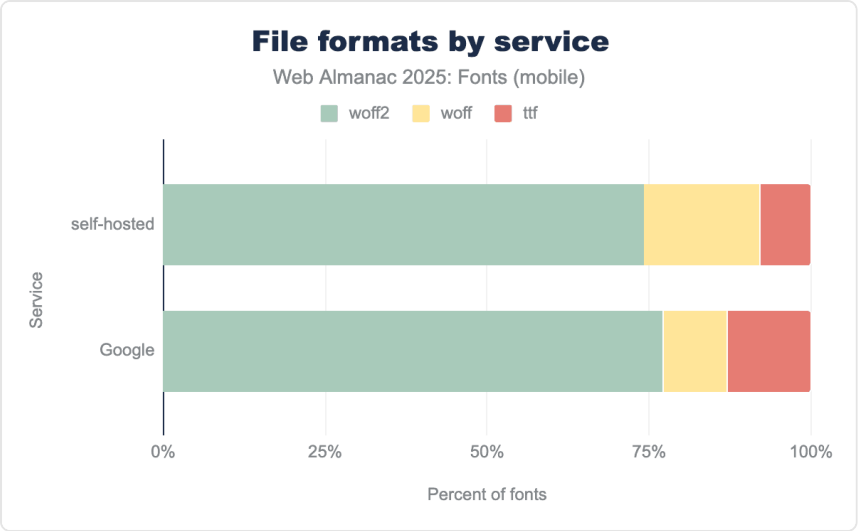


図1.11. サービス別のファイル形式（モバイル）。

2025年のデータは改善を示しています。より多くのセルフホスターがWOFF2に移行していますが、依然としてギャップがあります。セルフホストする場合は、フォントのWOFF2バージョンを確保し、正しい `Content-Type` でそれらを提供するようにサーバーを設定することが重要です。2024年にセルフホストのフォントファイルの大部分が設定が正しくないMIMEタイプで提供されていました（前述の5〜6%）。これは開発者がシンプルなサーバーの設定変更で修正できるものです。良いニュースは、WOFF2の優位性が全体的に拡大していることですが、Googleのプラットフォームよりもセルフホストの領域では少し遅いペースです。本質的には、セルフホストサイトは追いついていますが、最適なフォーマットの使用においてわずかに遅れています。

CSSテクニックとフォントの読み込み

CSSテクニックとフォントの読み込みは、タイポグラフィが実際に現実のものとなる場所です。ページはフォントを選択するだけでなく、フォントオリジンへの接続をいつ開始するか、どのファイルを最初にフェッチするか、転送中に何を表示するかをブラウザに伝えます。このセクションでは、それらのメカニズム（リソースヒント（`preconnect`、`preload`、そして消えゆく`dns-prefetch`））、テキストフォントとアイコンフォントの間で明確に分かれている現在の標準的な`font-display`パターン、そして基礎となるアウトラインフォーマットを見ていきます。

リソースヒント

ウェブフォントはレンダリングブロックやテキスト表示の遅延によって視覚的な異常を引き起こす可能性があるため、多くのサイトがフォントの読み込みを最適化するためにリソースヒントを使用しています。データは、ベストプラクティスの進化に伴っていくつかの変化はあるものの、これらのヒントの着実な採用を示しています。

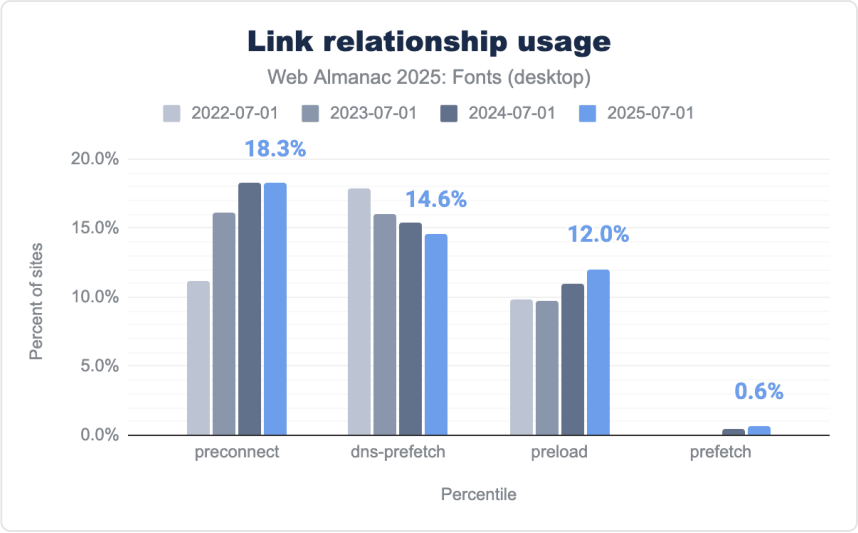


図1.12. デスクトップサイトでのフォントのリンクリレーションシップ値の使用状況。

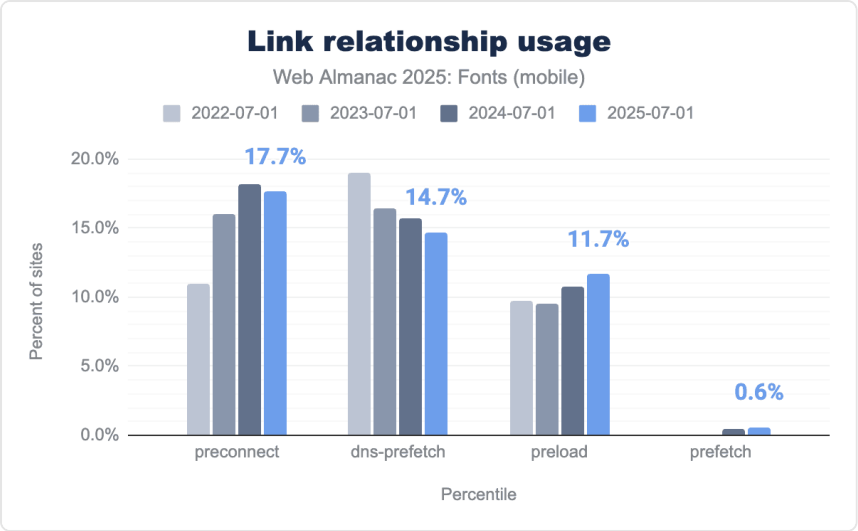


図1.13. モバイルサイトでのフォントのリンクリレーションシップ値の使用状況。

最も人気のあるヒントは依然として `<link rel="preconnect">` です（フォントホストへの接続を事前に開くためによく使用されます）。2025年のデータでは、デスクトップページの約18.3%がpreconnectヒントを含んでおり（2024年の18.3%からほぼ変わらず）、モバイルページの17.7%が同様です。この横ばいは、過去数年間増加した後、preconnectの使用が落ち着いたことを示唆しています。これは一般的なテクニックで、5ページに1ページ近くに存在しており、パフォーマンスを意識する多くのサイトがfonts.googleapis.comや自社CDNなどのドメインにpreconnectを追加するようになった現在、理にかなっています。

一方、`dns-prefetch`（DNSのみを解決する軽量なヒント）は低下傾向にあります。2025年には、フォントに`dns-prefetch`を使用しているデスクトップページが約14.6%で、2022年の約17.9%から減少しています。モバイルも14.7%で同様です。この減少は理にかなっています。開発者が`preconnect`（接続確立の一部としてDNS解決を含む）を発見すると、別個の`dns-prefetch`は必要性が低くなります。以前に`dns-prefetch`を使用していた多くのサイトが`preconnect`にアップグレードしており、これはTLSハンドシェイクを早期に行うことでより実質的な最適化を提供します。

preloadヒントは静かながら重要な上昇を見せています。2025年には、少なくとも1つのフォントに`<link rel="preload" as="font">`を使用しているデスクトップページが約12.0%で、2024年の約11.0%から増加しています。モバイルも同様の増加を示しています（昨年から約11.7%）。これは、ブラウザにフォントを発見させるのではなく、より多くのサイトが重要なウェブフォントファイルを明示的にプリロードしていることを示しています。フォントをプリロードすると、ブラウザがCSSを待ってからフォント宣言を待つのではなく、preloadヒントを見た瞬間にフォントのフェッチを開始するため、読み込み遅延を短

縮できます。このテクニックはパフォーマンスを重視したセットアップで特に人気があり、フォントファイルが大きい場合やFOIT（不可視テキストの点滅）を避けたい場合に推奨されます。

最後に、`prefetch`はニッチなツールのままです。2025年にはページのわずか約0.6%で使用されています（昨年の約0.5%からわずかに増加）。`Prefetch`は一般的に近い将来に必要になるかもしれないリソース（別のルートやページなど）を読み込むために使用されるため、`prefetch`が1%未満であることは、フォントに対するかなり専門的なヒントであることと一致しています。わずかな増加は、後続のページのアセットをプリフェッチするシングルページアプリケーションフレームワークや積極的なオプティマイザーによるものかもしれません。

まとめると、`preconnect`と`preload`が2025年で使用される主要なフォント読み込みヒントであり、`preconnect`がページの約5分の1に、`preload`が約8分の1に表示されています。これは`preconnect`に置き換えられるにつれての`dns-prefetch`の減少と一致しており、`prefetch`は稀ですが緩やかに成長しています。これは業界が、より強力でより明示的な読み込みシグナルへと移行していることを反映しています。開発者は投機的なもの（`dns-prefetch`、`prefetch`）よりも即時的なアクション（`preconnect`、`preload`）をますます好むようになっています。

フォント表示

CSS `font-display`ディスクリプタはウェブフォント使用において事実上の標準的なプラクティスとなっています。数年前は検討すべき「良い最適化」でしたが、2025年にはウェブフォントを使用するほとんどのページが、フォントの読み込み中にテキストがどのようにレンダリングされるかを制御するために font-display ポリシーを指定するようになっています。データは、テキストフォントとアイコンフォントの間で明確な分かれ目とともに、高速なテキストレンダリングへの強い選好を示しています。

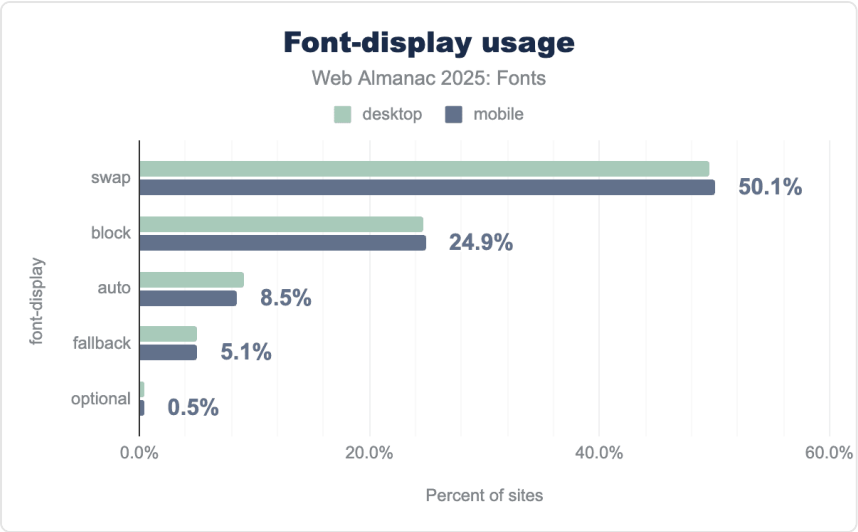


図1.14. font-display値の使用状況。

最も一般的な選択は `font-display: swap` です。2025年のクロールでは、デスクトップページの約49.6%、モバイルページの50.1%に表示されています。言い換えれば、ウェブフォントを使用するすべてのページの約半数が、少なくとも1つのフォントに `font-display: swap` を指定するようになっています。この値は、フォールバックフォントを使用してテキストを即座に表示し、ウェブフォントの読み込みが完了したときにスワップすることを保証します。 `swap` の人気はウェブのベストプラクティスと一致しています（Google Fontsや多くの専門家が推奨しています）。本質的に、テキストコンテンツのデフォルトの考え方になりつつあります。開発者は、一時的にフォールバックフォントであっても、テキストの素早い初回描画を好みます。

次に、 `font-display: block` はデスクトップページの約24.7%、モバイルページの24.9%で2番目に一般的な値で、 `swap` の約半分の普及率です。定義上、 `block` はフォントの読み込み中に一時的にグリフを非表示にしますが、これが主にボディテキストに使用されていると仮定すると危険に見えるかもしれません。しかし、データは実世界の `font-display: block` の使用の約70%がアイコンフォントスタイル（Font Awesome、テーマアイコンセットなど）からのものであることを示しており、作者が欠落したアイコンよりも短い遅延を意図的に好んでいます。それらのケースでは、 `block` はまさに期待通りのことをしています。最初のテキスト描画を最適化するのではなく、UIの破損を防ぎます。したがって、事実上すべての `block` の使用はボディテキストフォントからではなく、完全にレンダリングされるかまったくレンダリングされないかのどちらかが必要なUIアイコンからです。このパターンは2024年の章での「驚き」の発見でしたが、2025年のデータはそれを確認しています。 `block` の普及率は多くの開発者がコンテンツに不可視テキストを好むためではなく、アイコンキットが欠落したアイコンを防ぐためにデフォルトとして `block` と一緒に出荷さ

れているためです。

`swap` と `block` を超えると、他の値はずっと一般的ではありません。`font-display: auto`（ブラウザのデフォルト動作に委ねる）はデスクトップの9.1%、モバイルページの8.5%で使用されています。このシェアは、より多くの人が明示的に値を設定するにつれて縮小しています。また、`font-display: fallback`（非常に短い不可視期間を許可した後、常にスワップする）が約5.1%のページに見られます。この値は、テキストを素早くスワップしたいが、重要なフォントの読み込みに小さな余地を与えたい開発者に好まれることがあります。最後に、`font-display: optional` は依然として稀であり、ページの約0.4〜0.5%のみが使用しています。`optional` 値はパフォーマンスに対して最も積極的であり（フォントが即座に読み込まれない場合、まったくスワップしないことが多い）、通常はパフォーマンスに敏感なサイトや非常に特定のタイポグラフィのセットアップでのみ使用されます。

個々のフォントファミリーとそのfont-displayの動作を見ると、これらの結論が強化されます。最も一般的に読み込まれるテキストフォント（例：Roboto、Open Sans、Montserrat、Poppins、Inter、Lato）は、`@font-face` ルールで圧倒的に `font-display: swap` で提供されています。例えば、`swap` を持つRobotoは約12%のページに表示され、`swap` を持つOpen Sanは約8%に表示されます。逆に、`font-display: block` を持つフォントは主にアイコンキットです。Font AwesomeのCSS（デフォルトで `block` を使用）は約17%のページで観察され、ETmodules、IcoMoon、Bootstrap Iconsなどの他のアイコンフォントも一桁台の低いパーセンテージで `block` の使用に貢献しています。一部のテキスト指向のサイトは `font-display: fallback` を使用しています（例えば、Interやあるセリフフォントをするいくつかのサイトはスタイルなしフォントの点滅と偽ボールドテキストの点滅のバランスを取るために意図的に `fallback` を設定しています）。しかし、通常のテキストフォントに `block` を使用するサイトはほとんどありません。

現代のベストプラクティスは数字に明確に反映されています。ウェブの約半分が素早いテキストレンダリングを確保するために `swap` を使用し、`block` を使用する4分の1はほとんどが非テキストグリフのためにそうしています。残りの4分の1は `font-display` を指定していない（したがってブラウザのデフォルトを使用している）か、専門的な戦略（`fallback` / `optional`）を試みています。2024年と比較して、これらの割合は本質的に変わっていません。昨年は `swap` が約50%、`block` が約25%使用されていました。この安定性は2024年のガイダンスが依然として有効であることを意味します。サイトで `font-display: block` を見た場合、それはアイコンフォントである可能性が高いです。テキスト的なウェブコンテンツのほぼすべてについて、今日の開発者はより良いUXのために `swap` を好みます。

フォントアウトライン形式

タイポグラフィのもう一つの技術的な側面は、フォントファイル内のアウトライン形式（TrueType対PostScriptアウトライン）です。ほとんどのウェブフォントは、コンテナに関

係なく、TrueType（glyfテーブル）形式またはCFF（Compact Font Format）アウトラインのいずれかです。2022〜2025年のデータは、ウェブがTrueTypeアウトラインに圧倒的に標準化していることを示しています。

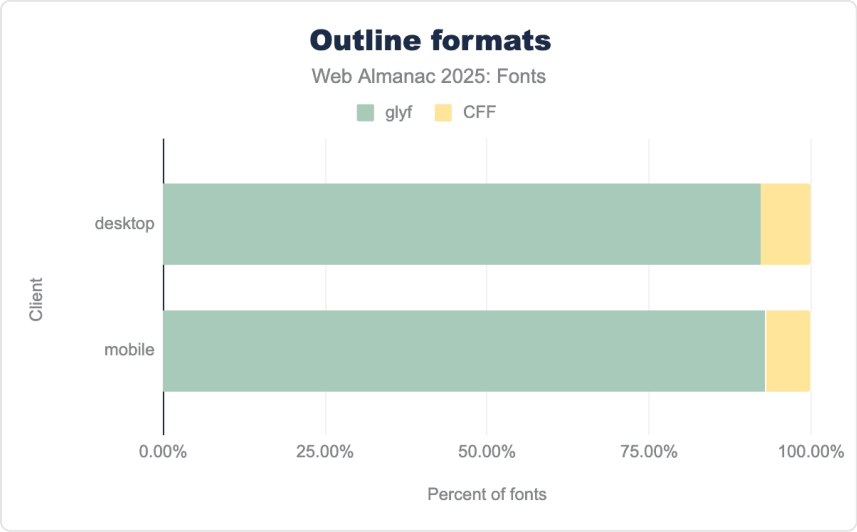


図1.15. アウトライン形式。

今年、ウェブに読み込まれたフォントの約92〜93%がTrueTypeアウトライン（伝統的な2次ベジェ曲線の「glyf」テーブル）を使用していました。これは2022年の約90%からわずかに増加しています。対照的に、PostScript/CFFアウトライン（`.otf`フォントによく見られる3次ベジェ曲線アウトライン）はウェブフォントの約7%に減少しています（数年前の約9〜10%から低下）。TrueTypeは特定のユースケース（特にバイトコードによるヒンティングを行わない場合）でパフォーマンスやサイズの利点があるため、モダンなウェブフォントサービスやファウンドリーは、元々PostScriptアウトラインとして設計されたフォントでもTTF/WOFF2を提供することが多いです。

その他はすべてごくわずかな割合です。CFF2（PostScriptアウトラインを持つ可変フォントに使用される新しいバリアント）は実際にはほぼ存在せず、フォントの0.01〜0.02%にしか表示されません。SVGグリフ（SVGテーブルを持つフォント、多くの場合カラーフォント）も非常に低い小数点以下の割合です。本質的に、カラー/絵文字フォントの特殊なケースを除いて、ウェブ開発者はこれらについてあまり心配する必要はありません。仮想的には常にTrueTypeベースのフォントを扱うことになり、特に古いパッケージからのCFFアウトラインが少数残っている程度です。

ハイフネーションとジャスティフィケーション

適切なテキストレイアウトは、読みやすさのために適切な場所で行を折り返すことを含みます。ブラウザはデフォルトでグリーディな改行アルゴリズムを使用しており、これはジャスティフィケーションされたテキストで不均一なスペースや「空白の川」を引き起こす可能性があります。代わりに、ハイフネーション（単語を分割する）と改行を改善するための新しいCSSプロパティ（`text-wrap: balance`または`pretty`）を使用できます。これらがどのように使用されているか見てみましょう。

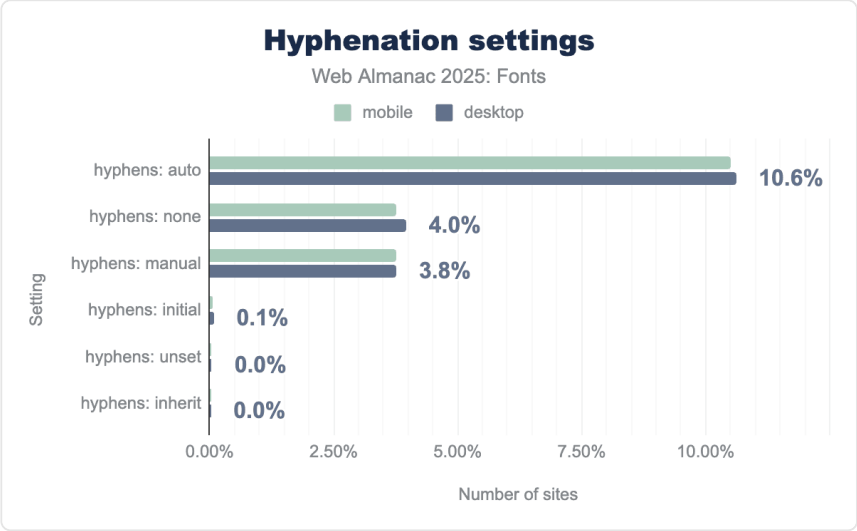


図1.16. ハイフネーションの設定。

まず、ハイフネーション（CSS `hyphens`プロパティ）について話しましょう。デフォルトでは、ブラウザは特定の条件が満たされた場合（`lang`が指定されていて単語が長いなど）にのみテキストをハイフネーションし、一部のブラウザ（Chromeなど）はそう指示されない限りまったくハイフネーションしないかもしれません。`hyphens: auto`プロパティは必要に応じて自動ハイフネーションを許可します。2025年には、世界的にページのわずか約10.5%がCSSで`hyphens: auto`を使用しています。これは過去数年よりわずかに増加していますが、依然として非常に低いです。これは、約10サイトに1サイトがハイフネーションを明示的に有効にしていることを示唆しています（通常、ジャスティフィケーションされたテキストや狭い列のテキストを改善するため）。

一方、`hyphens: none`または`hyphens: manual`が設定されているページが約3.8〜4.0%あります（多くの場合、リセットやフレームワークによって）。これは通常、自動ハイフネーションがないことを確保するために行われます（例えば、一部のデザイナーはタイトルや特定のコンポーネントでハイフンを避けることを好み、または古いCSSリセットが予期しな

い改行を避けるために `hyphens: none` を設定します）。

これらを組み合わせると、ページの約14〜15%が `hyphens` プロパティにまったく触れています。大多数（85%）はそれに触れておらず、事実上ブラウザのデフォルト（多くの言語でハイフネーションなしを意味する）に委ねています。したがって、ハイフネーションは普遍的にはほど遠く、おそらくそれを慎重に使用する必要があるため（正しい言語タグと様々なブラウザサポートへの期待を持って）、すべてのコンテンツがハイフネーションの恩恵を受けるわけではありません（これは主にジャスティフィケーションされたテキストや狭い列での懸念です）。

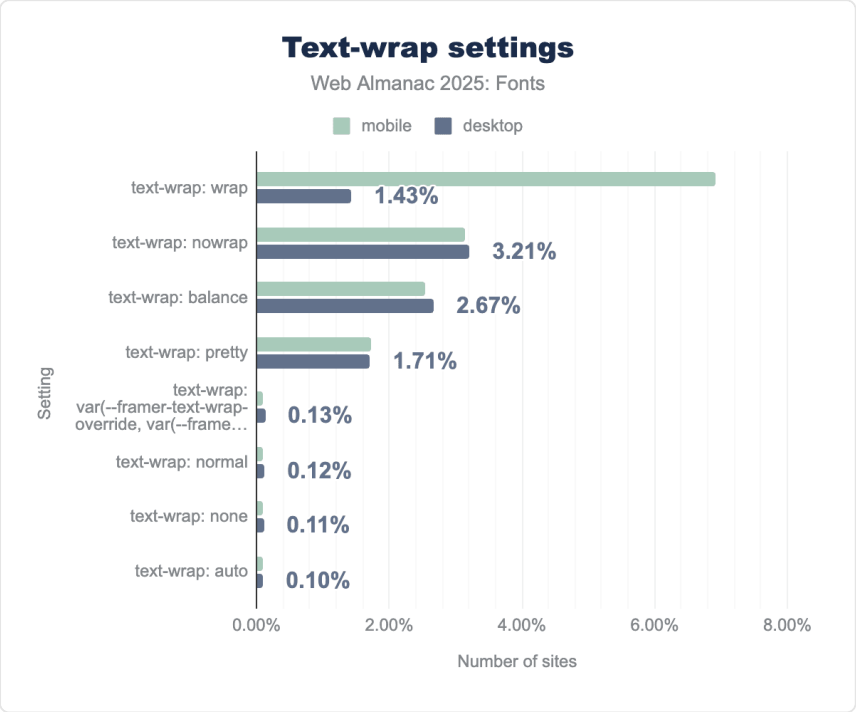


図1.17. テキストラップの設定。

テキストの折り返しと改行（CSS `text-wrap` プロパティ）：CSS Text Module Level 4は、ブラウザがテキストをより均等に分配するのを助けるために `text-wrap: balance` と `text-wrap: pretty` を導入しました。これらはかなり新しく（SafariやChromiumベースのブラウザなど一部のブラウザでのみ最近サポートされています）、その採用は小さいながらも、複数行の見出しなどのための新興のベストプラクティスを示しています。

デスクトップサイトでは、ページの約2.7%が `text-wrap: balance` を使用しています。モバイルでは約2.5%です。このプロパティは行を概ね等しい長さにしようとします（複数行の

タイトルに最適）。その新しさにもかかわらず、約40ページに1ページですでに確認できるのは印象的です。これはフレームワークや大規模サイトがヒーローセクションへの素早い採用を反映しているものと思われます。

`text-wrap: pretty` プロパティ（非常に短い単語や他の奇妙な配置を避けるためにスペースにある程度の余裕を持たせる）は、ページの約1.7%（デスクトップとモバイルの両方）にあります。これは `balance` プロパティよりわずかに低いです。 `Pretty` はより段落向けで、厳しい改行を避けるためのものであり、完全なジャスティフィケーションや特定の改行の問題が発生する場所で選択的に使用される場合があります。

興味深いことに、モバイルでは `text-wrap: wrap`（デフォルトの通常の動作）が約6.9%のページで明示的に表示されています。これはフレームワークが再述しているか、小さい画面のために `balance` から `wrap` にトグルしているためかもしれません。そして `text-wrap: nowrap`（要素内での折り返しを防ぐ）は、ボタンやロゴなどを1行に保つためによく使用され、モバイルページの約3.1%、デスクトップページの3.2%で使用されています。

モバイルでの `nowrap` と明示的な `wrap` の比較的高い使用は、小さい画面での改行の制御が特に重要であることを示しています（例えば、タイトなUI要素での不自然な折り返しを防ぐ）。

ハイフネーションとテキストラップの機能を合わせると、次のような状況が浮かび上がります。少数のサイトがテキストレンダリングのきめ細かな制御を本当に行っており（これらはニュースサイト、慎重に設計された編集サイト、またはアプリのようなインターフェースが含まれる可能性があります）、それらのサイトはハイフネーションとバランスの取れた折り返しを使用してテキストブロックを改善しています。大多数のサイトはデフォルト（ハイフネーションなし、グリーディな改行）に依存しています。

ジャスティフィケーションについて具体的に言うと：サイトが `text-align: justify` を設定した場合（一部のサイト、特に記事コンテンツでそうしています）、理想的には大きなギャップを避けるために `hyphens: auto` とおそらく `text-wrap: pretty` も使用すべきです。まだ多くのサイトがそうしていないことがわかっており、ウェブ上のジャスティフィケーションされたテキストの多くはおそらく最適でないスペースを持っています。ツールは存在します（ハイフネーション、 `balance` 、 `pretty` ）が、採用は遅れています。その一部はブラウザのサポート（これらはある程度最先端のプロパティです）であり、一部は認知不足かもしれません。

最後に、 `hyphenate-character`（カスタムハイフングリフを指定する）や `hyphenate-limit-chars`（非常に短い単語のハイフネーションを避ける）などのプロパティはほぼ未使用のようです。これらのプロパティの使用は無視できるほど少なく、ほとんどの著者が触れない非常に細かい調整です。ほとんどの著者はブラウザのデフォルトを受け入れています。

したがって、ハイフネーションは緩やかに成長していますが、依然として約10サイトに1サ

イトしか使用しておらず、新しい改行オプションが登場しています。数パーセントのサイトがすでにバランスの取れた改行を使用して、より良いタイトルを表示しています。レスポンシブデザインと複数行のレイアウトテクニックが開発者によりよい制御を求めるよう促すにつれて、これらの数字が増加することが予想されます。

ファミリーとファウンドリー

このセクションではウェブタイポグラフィの供給側を見ていきます。まず実際にCSSで宣言されているフォントとファウンドリーを追跡し、次にシステムフォントと汎用スタックの役割を見て、最後にメタデータをファウンドリー、デザイナー、ライセンスまで追って、ウェブ全体のユーザーに最も普及しているものを確認します。

人気のフォントファミリー

ウェブデザイナーは実際にどのフォントファミリーを使用しているのでしょうか？ページのCSSを調べることで、どのfont-family名が最も頻繁に宣言されているかを確認できます。2025年の結果は、ユビキタスなアイコンフォント、広く使用されているウェブフォント、システムフォントフォールバックの組み合わせを示しています。

CSSで見られる最も一般的な単一のフォントファミリーは依然としてFont Awesome（主要なアイコンフォント）です。データセットのページの約8.5〜9.3%で宣言されています。この高い普及率は、多くのサイトがFont AwesomeのCSS（一部のアイコンしか使用していなくても）を含んでいることと、一部の人気フレームワークがデフォルトでバンドルしていることによります。Font Awesomeがトップにいることは、インラインSVGアイコンが引き続き普及するにつれて変わるかもしれませんが、開発者がUIアイコンにアイコンフォントを頻繁に依存していることを強調しています。

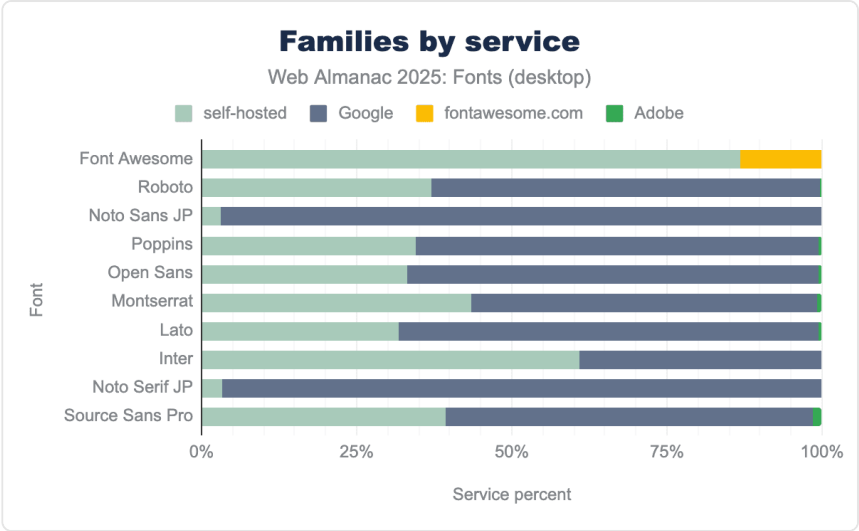


図1.18. デスクトップサイトでのサービス別フォントファミリー。

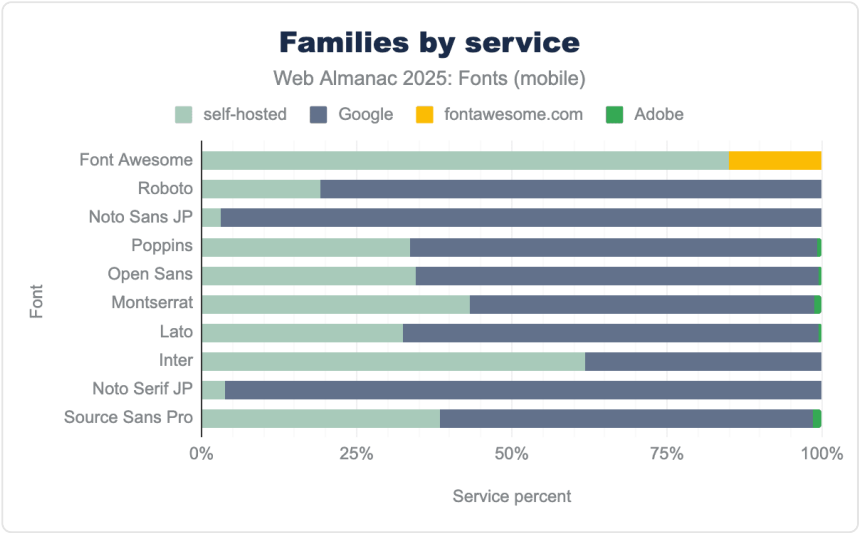


図1.19. モバイルサイトでのサービス別フォントファミリー。

その後ろにはGoogle Fontsや同様のライブラリからの通常のテキストワークホースが続きます。Robotoはページの約10〜10.7%で宣言されており、全体で2番目に人気のあるファミリーとなっています。Poppins（ページの5.8〜6.0%）とOpen Sans（5.0〜5.6%）が続きます。これらのGoogle Fontsの定番はウェブでのサンセリフボディとUIテキストのトップチョイスです。次はMontserrat（ページの約3.3〜3.9%）で、近年見出しやブランディングで人気が

高まっています。注目すべき上昇株はInterで、現在ページの約1.4〜1.5%に使用されています。InterはUIデザイン用の高く評価されたオープンソースフォントであり、その使用率の上昇は多くのモダンフレームワークやデザイナーによる採用を反映しています。Oswald、Nunito、Ralewayなど他の馴染みのある名前もページの数パーセントに表示されています。

主な学びは、アイコンフォントと少数の汎用性の高いサンセリフがCSSのランドスケープを支配していることです。CSSでのFont Awesomeの存在感（使用するすべてのサイトがエンドユーザーにアイコンをレンダリングしているわけではないにもかかわらず）は、フレームワークがどのように採用を促進できるかを示しています。テキストフォントの中では、Googleのオープンソースの提供（Roboto、Open Sans、Montserrat、Poppins、Latoなど）が大きなマージンでリードし続けています。これらは無料で利用でき、広く知られており、多くの言語をカバーしているため、驚くべきことではありません。また、新しいオープンソースフォント（Interなど）がニーズを満たすとき（Interの場合、スクリーン用に最適化され、機能が非常に充実）、依然としてメインストリームに入り込めることも見て取れます。

汎用フォントとシステムフォント

カスタムウェブフォントの普及にもかかわらず、システムフォントと汎用ファミリー名は特にパフォーマンスとネイティブな外観のためにウェブデザインの基盤であり続けています。多くのサイトがボディテキストやUI要素にシステムフォントスタックまたはフォールバックを指定しています。2025年のデータはこれらがいかに一般的かを示しています：

ファミリー	デスクトップ	モバイル
sans-serif	89%	89%
monospace	64%	63%
serif	47%	48%
system-ui	10%	9%
ui-monospace	4%	4%
ui-sans-serif	4%	4%
cursive	3%	3%

図1.20. 最も一般的なシステムファミリー

汎用の `sans-serif` はCSSで最も一般的なファミリー名（font-familyリストの末尾のフォールバックとして）です。デスクトップとモバイルページの約89%に含まれています。本質的に、ほぼすべてのサイトがCSSに常にデフォルトフォントを確保するために `sans-serif` で

終わる（またはシステムフォントスタックでそれから始まる）行を持っています。汎用の `monospace` も非常に一般的で（ページの約64%に見られる）、開発者がコードスニペットや整形済みテキストにモノスペースフォントを設定することが多いためです。汎用の `serif` はページの約48%に表示され、見出しのフォールバックとして、またはセリフデザインが必要な場合に使用されます。

より興味深いのは、`system-ui` のようなプラットフォーム固有の汎用フォントの台頭です。`system-ui` 値（macOS/iOSのSan Francisco、WindowsのSegoe UI、AndroidのRobotoなど、OSのUIフォントにマッピングされる）は現在、デスクトップページの約9.7%、モバイルの9.3%で使用されています。これは著しい増加であり、数年前はこれらの値はほとんど登録されていませんでした。採用は主に「ネイティブアプリ」の美学を目指すか、パフォーマンス上の理由からユーザーのデフォルトUIフォントに委ねるモダンなCSSフレームワークやデザインシステムによって促進されています。同様に、新しい汎用の `ui-monospace`（システムのデフォルトのモノスペースフォント）はページの約4%で使用され、`ui-sans-serif` は約3.5%で使用されています。これらの数字は、（約10サイトに1サイト）の無視できないサブセットのサイトが、多くの場合スピードとネイティブアプリとの一貫性のために、テキストにシステムフォントを明示的に選択していることを示しています。これはGitHubやMediumなどのサイトがボディテキストにウェブフォントを読み込まないようにシステムフォントを使用するトレンドと一致しています。

まとめると、システムフォントは多くのウェブサイトで典型的に縁の下の力持ちでした。開発者はほとんどのUIテキストにシステムフォント（または少なくとも `sans-serif` のような安全な汎用フォールバック）を使用し、次にブランディングやヘッダーのためにいくつかのカスタムフォントを読み込むというハイブリッドアプローチを使用することが多いです。汎用の `sans-serif`、`serif`、`monospace` はフォールバックとしてユビキタスであり、`system-ui` ファミリーと関連するものは現在多くのCSSフレームワークでより目立っています。

フォントファウンドリー

フォントの作成に功績があるファウンドリーやベンダーを確認するために `@font-face` メタデータを見ると、大多数は少数のファウンドリーによって提供されており、クレジットのないフォントの大きな塊と、低いパーセンタイルで長いテールの他のファウンドリーが続いています。

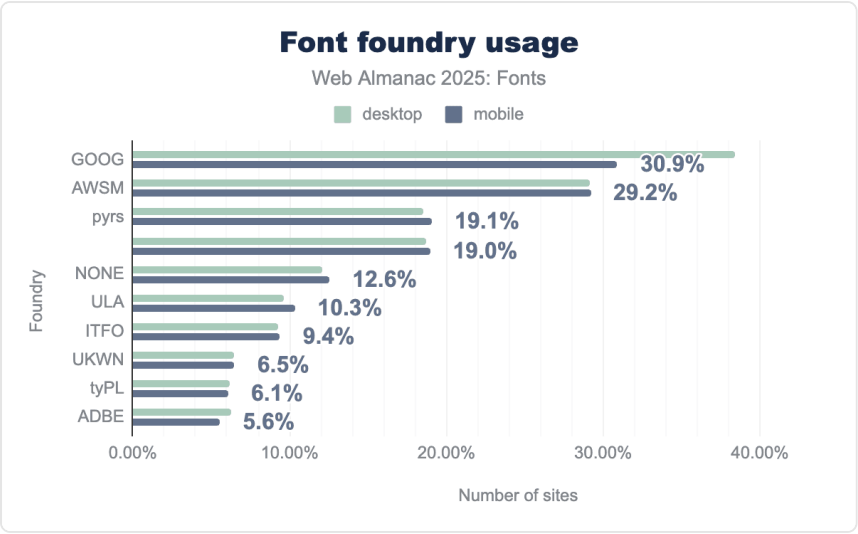


図1.21. フォントファウンドリーの使用状況。

ページカバレッジ別で、ウェブのトップの「ファウンドリー」ラベルは：

- **Google**：ウェブフォントを使用するウェブサイトの約34%でファウンドリーとして記載されています。これにより、Google（Google Fonts経由）は断然フォントの最大の単一ソースとなっています。これは主に独立したデザイナーによるオープンソースデザインへのGoogleの資金提供によるものです。

- **Font Awesome（AWSM）**：ウェブサイトの約29%でクレジットされています。これは2番目に大きなシェアで（Googleより約5パーセントポイント後れを取っており）、Font Awesomeアイコンフォントのユビキタスさによって完全に説明されます。Font Awesomeはアイコンセットであるため（ファウンドリー「Dave Gandy」も引用される可能性がありますが、フォントメタデータはよく「Font Awesome」または「AWSM」をファウンドリーとして使用します）、少しユニークです。

合わせると、GoogleとFont Awesomeに関連するフォントは全ページの約3分の2に表示されており、これは巨大です。これら2つの後、次に大きなカテゴリは実際には帰属のないフォントまたはカスタムフォントです。ページの約19%がファウンドリーが指定されていないフォント（「UNSPEC」）を持ち、別の12%がファウンドリーに「none」または汎用的な値を記載しています。合計すると、フォントの4分の1をはるかに超えるものがメタデータに明確

なファウンドリー情報を持っていません。このチャンクはカスタムのセルフホストフォント、小さなファウンドリー、または名前が標準化されていなかったフォントを表している可能性が高いです。

GoogleとFont Awesomeを超えた識別可能なファウンドリーの中では、いくつかが際立っています。「pyrs」（Fontlab）はページの約17%に記載されています（これはオープンソースのアイコンセットや特定のCJKフォントに関連するラベルで、複数のケース変化から正規化が必要かもしれません）。ULA（MontserratのデザイナーであるJulia Ulanovsky）は約10%です。Indian Type Foundryは約9%です。ADBE（Adobe）は約6%です。これらの数字は昨年と同程度の大きさで、Notoなどのフォントがどのように帰属されるかによって若干の順序変更があります。例えば、NotoフォントはGoogleと他者のコラボレーションであったため、これらの共同フォントの一部がAdobeのファウンドリーコードの下に表示されるようになり、昨年の計算方法と比較してAdobeのシェアがわずかに増加しています。

要するに、ウェブ上のフォントへの帰属は現在、Google、Font Awesome、数十の人気フォントの発行者、そして匿名の大きな中間層に分かれています。約30%のフォントがファウンドリー情報が不明確または存在しません。残りはアクティブなウェブフォントファウンドリーの「who's who」に分かれています。Indian Type Foundry（Google経由で多くの人気フォントをリリース）、「pyrs」などのファウンドリータグにグループ化された独立したデザイナー、Adobe、Monotypeなどの商業ファウンドリーの少々などです。この分布はオープンソースフォントのプロバイダーとしてのGoogle Fontsの影響力とFont Awesomeの広範な使用を強調しながら、ウェブフォントの大部分が帰属が強くない、より広いオープンソースコミュニティやカスタムキットから来ていることを思い起こさせます。

フォントデザイナー

多くのフォントファイルはメタデータにタイプデザイナーの名前を含んでいます。それを集計することで、誰の作品がウェブで最も普及しているかを確認できます（多くのフォントがメタデータにデザイナーをまったく記載していないことを念頭に置いて）。

デザイナー	デスクトップ	モバイル
	78.3%	76.7%
Dave Gandy	13.6%	13.8%
Adrian Frutiger	1.5%	1.6%
Jan Kovarik	1.5%	1.4%
Rasmus Andersson	1.2%	1.2%
Linotype Design Studio	1.0%	1.2%
Google	1.1%	1.1%
Julieta Ulanovsky	1.0%	1.1%
Mark Simonson	0.9%	0.8%
Commercial Type, Inc.	0.4%	0.7%

図1.22. 最も人気のあるフォントデザイナー

データは、ほとんどのウェブフォントがデザイナーを指定していないことを示しています。ページの約77〜78%がメタデータにデザイナーが記載されていないフォントを少なくとも1つ使用しています。これは省略によるものか、フォントが個人ではなくファウンドリーやプロジェクトに帰属しているためかもしれません。その巨大なブランクカテゴリは、「デザイナーランキング」を割り引いて見る必要があることを意味します。これは部分的な見方です。

そうは言っても、クレジットされたシェアでは一人の名前が際立っています。Dave GandyはFont Awesomeの作成者であり、その名前が約13.7%のページに表示されています（基本的にFont Awesomeをセルフホストしてデザイナーフィールドが入力されているすべてのページ）。これにより彼はアイコンフォントの普及によって、ウェブ上で最も目立つ（メタデータ上の）タイプデザイナーとなっています。興味深い特異性です。ウェブ開発者が美的な理由から意識的に「Dave Gandyのタイプフェイス」を選んでいるわけではなく、一つの非常に成功したツールキットが何百万ものサイトに彼の名前を付けているのです。

その他には、それぞれ約1%前後のページに名前が登場する著名なタイプデザイナーたちのグループがいます。Univers やエポニマスなFrutiger書体で知られるAdrian Frutiger（1.5%）、InterのデザイナーであるRasmusAndersson（1.2%）、汎用的なラベル「Google」（1.1%、特定のデザイナーがクレジットされていない一部のGoogleフォントで使用）、Linotypのデザインスタジオ（1.1%）、Montserratのデザイナーである Julieta Ulanovsky（1.0%）、Proxima Novaなどのデザイナーである Mark Simonson（0.9%）などです。

これらの数字はウェブで最も人気のあるフォントを反映しています。MontserratはUlanovskyの名を持ち、InterはAnderssonの名を持ち、Proxima NovaはSimonsonの名を持っています（ただし、Adobe経由のProximaはキットのメタデータによっては彼の名前が記載されない場合もあります）。さらにロングテールでは、Łukasz Dziedzic（Latoのデザイナー、おそらく複数の綴りで登場）やGoogleフォントプロジェクトや主要なファウンドリーへの貢献者の名前も見られます。しかし、これらのシェアはそれぞれ1%以下です。

重要な点は、ウェブ上でのデザイナー帰属は非常に疎らで偏りが激しいということです。1つのアイコンフォントの作者がツールの人気により統計を圧倒しています。残りの見えるデザイナー帰属は、著名な20世紀タイプデザイナー（Frutigerなど）、最も人気のあるオープンソースフォントの作者（InterのAndersson、MontserratのUlanovsky）、大きなファウンドリーチーム（Linotype、Monotypenど）が混在しています。これはウェブでのフォント配信のあり方を示しています。多くの場合、プラットフォーム（Googleなど）やファウンドリーが前面に出て、個々の作者は名前の残る大きなプロジェクトに携わっていない限り、常に名前が挙げられるとは限りません。

フォントライセンス

フォントファイルにはメタデータにライセンスのURLや識別子が含まれていることが多く、ウェブで一般的なライセンスを把握するために利用できます。ただし、約半数のフォントはパース可能な形式でライセンスを明確に指定していません。それを念頭に置くと、データはオープンソースライセンスがウェブフォントを完全に支配していることを示しています。

ライセンス	デスクトップ	モバイル
OFL	64%	65%
	49%	50%
Font Awesome	13%	13%
Apache	21%	8%
Adobe	5%	4%
Monotype	4%	4%

図1.23. 最も一般的なフォントライセンス

断然最も一般的なライセンスはSIL Open Font License（OFL）です。サンプル中の約64%のウェブサイトがOFLのもとで少なくとも1つのフォントを使用しています。これはOFLがGoogle Fontsやその他のオープンソースフォントプロジェクトの大多数で使用されているライセンスであることを考えれば驚くべきことではありません。基本的に、変更時に名前を変更する限り、フォントの自由な使用、変更、再配布を許可しています。OFLフォントの普及

は、ほぼ3分の2のサイトがオープンソースフォントを使用していることを意味します。

次の大きな「ライセンス」カテゴリは、実際にはライセンスが指定されていないものです。約50%のページで、少なくとも1つのフォントのライセンスフィールドが空白または認識不能でした。これはフォントがメタデータのないカスタムフォントであるか、ライセンスが埋め込まれていない（よくあること）商用フォントであるか、あるいは「不明」として入力されているデータである可能性があります。ウェブ上のフォント使用の多くは、ファイル内のライセンスによって簡単に追跡できないことを思い起こさせます。しかしこのバケットの多くはおそらく、`@font-face` にライセンスを宣言しない（ウェブデザイナーが常に記入するとは限らないため）専有フォントです。

Font Awesomeライセンス（専有とオープンのハイブリッドライセンス）は約13%のページに存在し、Font Awesomeの使用数と一致しています。Adobeのフォントライセンス（Adobe Fonts経由のフォント用）は約4〜5%のページに表示され、これもAdobeの使用シェアと一致しています。同様に、MonotypeのEULAは約4%をカバーしています（ただし、ウェブ上の多くのMonotypeフォントはセルフホストされている場合「ライセンスなし」として表示される可能性があります）。

それ以外に、Apacheライセンス（一部の古いGoogle Fontsやその他のプロジェクトで使用されているオープンソースライセンス）はOFLに次ぐ2番目に多い明示的なライセンスであり、オープンライセンスが標準であることを強調しています。興味深いことに、デスクトップ/モバイルの差があります。デスクトップのクロールでは高く（〜21%）、モバイルでは低く（〜8%）なっており、特定のフォントやフレームワークがデスクトップサイトでより一般的であることが原因と考えられます。

これらを超えると、特定のファウンドリーとベンダーのライセンスの長いテールがあります。Commercial Type、Typotheque、Hoefler&Co（Typography.com）、Dalton Maag、Naver（一部のNoto Sans CJKディストリビューション用）のマーカーがそれぞれ1%未満のページで見られます。これらは特定のサイトでの商用フォントライブラリの特定の使用を示しますが、概してウェブはオープンフォントライセンスで動いています。

グローバル言語サポート

ウェブはますますグローバルになっており、ウェブのタイポグラフィも世界の多くの書記体系をサポートしなければなりません。ラテン文字はウェブフォントで最もよくサポートされている書記体系ですが、近年のフォント作成と配布の取り組みのおかげで、他の書体も着実に増加しています。

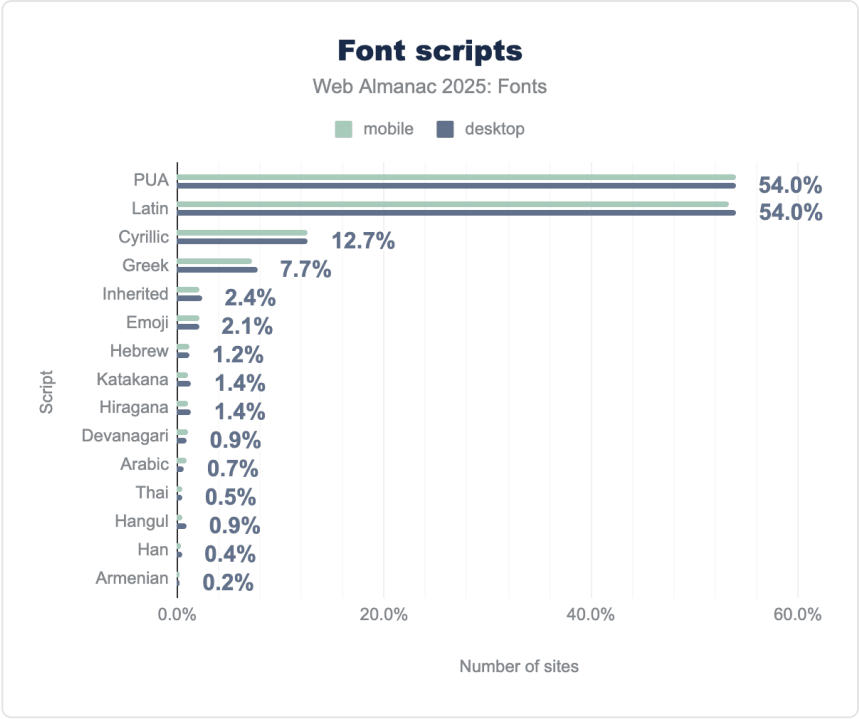


図1.24. フォント書記体系の使用状況。

2025年には、全ウェブフォントの半数強にラテン文字が含まれています（データセットのフォントの約54%）。ラテン文字はほとんどのマルチスクリプトフォントに含まれているため、これは基本的にヨーロッパやアメリカのオーディエンス向けのフォントがすべてここに含まれることを意味します。ラテン文字と並んで、約54%のフォントにはUnicodeの私用領域（PUA）も含まれており、これはアイコンフォントやカスタムシンボルが存在する場所です。PUAの普及は、ウェブ上のフォントの多くが実際にアイコンセットであるか、組み込みのアイコンを持っているか（Font Awesome、テーマアイコンフォントなど、すべてPUAにマッピングされる）を（再度）示しています。

ラテン文字の次に多い書体はキリル文字で、ウェブフォントの約12〜13%がサポートしています。キリル文字のシェアは少し成長しており（2022年は約7%、2024年は約10%）、ロシア語、セルビア語、ウクライナ語などの言語をカバーするフォントが増えていることを反映しています。ギリシャ語のカバレッジは約7〜8%のフォントに存在し、数年前から増加しています。多くの拡張文字セットはラテン文字、ラテン拡張、キリル文字、ギリシャ語を1つのファイルに持っているため、これらの拡張はしばしば一緒に来ます。

CJK（中国語・日本語・韓国語）書体の存在は見えますが、主にファイルサイズの懸念から

人口規模が示唆するよりも控えめです。日本語書記体系はカタカナ（約1.3〜1.4%のフォントにカタカナ文字が含まれる）とひらがな（同様に約1.3〜1.4%）で登場します。さらに、漢字（中国語と日本語の書き込みに使用される文字）は約0.4%のフォントに登場します。漢字の割合が小さいのは、フルの中国語/日本語フォントファイルは膨大であり、比較的少数のサイトがそれらを完全に提供しているためです。多くの場合、サイトはシステムフォントに頼るかサブセットを使用しています（カタカナ/ひらがなの存在は一部の日本語フォントやサブセットが使用されていることを示唆しています）。韓国語のハングル文字は約0.9〜1.0%のフォントに含まれています。これらの数字は小さく見えますが、一部のオープンソースCJKフォント（Noto Sans JP、Noto Sans KRなど）が使用されていることを表しています。サイズのためにほとんどのウェブサイトがCJKウェブフォントを提供していなかった10年前と比較すると、これは大きな変化です。

注目すべき存在の他の書体：ヘブライ語は約1.2%のフォントに登場します（Google経由とセルフホストの両方でいくつかのヘブライ語ウェブフォントがあります）。アラビア文字はデスクトップで約0.7%、モバイルで0.9%のフォントがサポートしており、モバイルの多い地域がウェブフォントを多く使用するにつれてわずかに増加しています。デバナーガリー（インドのヒンディー語やその他の言語に使用）は1%弱のフォントに存在します。タイ文字は約0.5%のフォントに見られます。アルメニア語とジョージア語はそれぞれ0.2〜0.3%の範囲です。グジャラート語、グルムキー語、タミル語、ラオス語、クメール語などの他の南アジアおよび東南アジアの書体はそれぞれ0.1%未満のフォントに登録されています。

ラテンアルファベット以外の多くの書体が2%未満のフォントで使用されていますが、総合的には広い採用を示しています。主要なトレンドは、ラテン文字のみのフォントの優位性がゆっくりと低下していることです。より多くのフォントがマルチスクリプトであるか、他の書記体系を対象としています。このグローバル言語サポートの拡大の多くは、オープンソースフォントプロジェクトのおかげです。例えば、日本語と韓国語の重要な使用は、高品質のCJKフォントを自由に利用可能にしたNoto Sans/Serif JP、Nanum Gothic、Noto Sans KRなどのオープンフォントに起因しています。2024年には、韓国語のトップ10ウェブフォントファミリーのすべてがオープンソースであることに気づきましたが、そのパターンは続いており、デザイナーが自分の言語のための無料の高品質フォントにアクセスできるとき、それらのフォントを使用することを示しています。同様に、日本語のトップウェブフォントはオープンソース（Notoなど）であり、共有された漢字のために隣接言語に多大な影響を持っています。

1つの注意点は中国語（特に簡体字中国語）です。中国語専用のウェブフォントが使用されているのを見ることは依然として稀です（上記のデータは日本語サブセット以外での漢字の使用が非常に低いことを示しています）。何千もの文字を含める場合、中国語フォントファイルは極めて大きいため、ほとんどのサイトはまだそれらをセルフホストすることを避けています。システムフォントを使用するか、ロゴや見出し用に非常に限られたサブセットを提供しています。したがって、日本語と韓国語のウェブフォントの成長が見られる一方で、中国語は主にパフォーマンスの理由から同じ普及を見ていません。これからの発展の予告として、インクリメンタルフォント転送（IFT）の開発は、過去に中国語ウェブフォントを妨げ

てきたパフォーマンスダイナミクスを変えることを約束しています。IFTにより、ユーザーは特定のページを表示するために必要な文字のみを取得できるようになり、大きな文字セットを持つ言語のファイルサイズが劇的に削減されます。標準化され広く採用されれば、IFTはそれらの大きな文字セットの実効コストを削減することで、フル機能の中国語ウェブフォントをはるかに実用的にする可能性があります。同じメカニズムは、IFTが未使用のグリフと未使用のバリエーションデータの転送を避けることができるため、大規模な多軸可変フォントにも広く利益をもたらし、豊富な可変タイポグラフィをスケールでデプロイする際のコストを削減します。

まとめると、ウェブフォントの景観は徐々に多言語的になっています。ラテン文字はまだフォントの約半分に存在し（ほぼすべてのサイトは、数字や基本記号だけのためでもラテン文字が必要です）、他の書体をサポートするフォントのシェアは年々拡大しています。これは地域化の増加と、十分に提供されていない言語のフォントを提供してきたGoogleのNotoプロジェクトなどの取り組みの成果のポジティブなサインです。開発者にとって、例えばデバナーガリーやアラビア語のフォントが必要な場合、オープンウェブフォントが存在し、同僚によって使用されている可能性が相当あることを意味します。また、マルチスクリプトフォントは膨大になる可能性があるため、IFTなどの実験的な開発からフォントファイルサイズとパフォーマンスに注意が必要であることも意味します。

高度なフォーマットと機能

ウェブフォントがより多くの書体とより大きな文字セットをカバーするために拡張されるにつれ、次の問題はこれらのフォントで実際に何ができるかということです。この最終セクションでは、基本的なカバレッジを超えて、現代のフォントファイル内の高度な仕組み—OpenType機能、可変フォント、カラーフォント—と、開発者がウェブ全体でそれらの機能をどのくらいの頻度で使用しているかを見ていきます。

OpenType機能

OpenType機能は、印刷時代のタイポグラフィクラフトの特徴となっていた合字、代替文字、古いスタイルの数字、その他の詳細などの高度なビジュアルスタイルを可能にします。これらの機能の採用を2つのアプローチから観察できます。フォントにOpenType機能が含まれているかどうか、そしてサイトがCSS経由でそれらの機能を使用しているかどうかです。

まず、フォント自体を見ると、OpenTypeレイアウト機能はウェブフォントで今や標準となっています。2022年のデータでは、OpenTypeレイアウトテーブル（GSUB/GPOS）を持つフォントは半数未満でした。2024年には小さな多数（モバイルで約54%）になりました。2025年には、ウェブ上の個別のフォントファイルの約61〜62%が少なくとも1つのOpenType機能テーブル（置換や配置など）を含んでいます。この使用レベルはフォントの明確な多数を表しており、毎年成長しています。これは新しいフォントファイル（特にオー

プンソースのもの）がフル機能を含んで構築されているのに対し、古いシンプルなフォント（またはアイコンフォント）はそれらを持っていない可能性があることを反映しています。

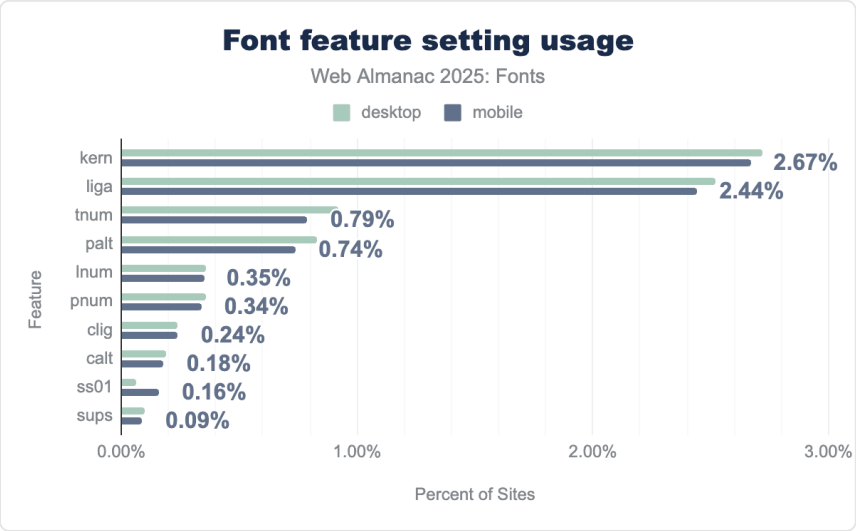


図1.25. 機能別のCSS font-feature-settingsの使用状況。

OpenType機能の採用を使用量（フォントリクエスト）で重み付けすると、シェアはさらに高くなります。リクエストベースでは、フォントフェッチの99%以上がOpenType機能を持つフォントに対するものです。この数字は、OpenType機能を持たないフォントがニッチまたは非常に低使用（または多くのファイルとしてカウントされるが、トラフィックのシェアが小さいアイコンフォント）である傾向があることを示しています。基本的に、ユーザーがウェブで実際に見るほぼすべてのフォントは、合字やカーニングなどの機能が可能です。サイトがこれらの機能を有効化していなくても、フォントファイルはそれらをサポートしています。

したがって、多くの開発者は使用しているフォント（特にテキストフォント）でOpenType機能が利用可能だと想定できます。これらのテーブルを欠いているのは少数派（10個のフォントのうち4個未満、縮小中）で、ほとんどが古いウェブセーフに近いフォントや特殊なケースです。

では、フォントでどのOpenType機能が一般的でしょうか？OpenType機能を持つフォントの中で、最も一般的なものは：

- **カーニング（`kern`）**: 約44〜46%のフォントにカーニング機能が含まれています。数年前は〜35〜38%に過ぎなかったため、その存在は大幅に増加していま

す。多くのフォントが特定の文字ペア間のスペーシングを調整するためにGPOSカーニング（またはkernテーブル）に依存していることを意味します。品質と活版整理の洗練さが標準になっているサインです。

- **標準合字（`liga`）**: 約43%のフォントがこの機能を含んでいます。これは文字のシーケンス（「fi」など）を単一の合字グリフに置き換えます。2022年の約10%からのこの大きな跳躍は、ほとんどの新しいフォントが合字をサポートしていることを示しています（基本的なものだけでも）。

- **ローカライズされたフォーム（`locl`）**: 約33%のフォントがこれを持っており、異なる言語で異なるグリフを許可します（ポーランドのkresakアクセントやトルコのドット付きiなど）。これが3分の1であることは、多くのマルチ言語フォントが組み込みのローカル最適化を持っていることを意味します。

- **数値機能**: 約33%が分数（frac）をサポートし、約25%が分子（numr）と分母（dnom）をサポートしています（任意の分数の構築用）。表形式数字（tnum）と比例数字（pnum）はそれぞれ約20%のフォントに含まれています。これは素晴らしいことです。ランダムな現代のフォントを選ぶと、テーブル、財務データなどに役立つ高度な数字の整列と分数構築能力を持っている可能性が相当あることを意味します。

- **マーク配置（`mark` と `mkmk`）**: 約17%が基本的なマーク配置を持ち、14%がマークからマークへの配置を持っています。これらはアラビア語、インド系書体、ベトナム語文字などのアクセントスタッキングにとって重要です。約6分の1のフォントにマーク配置が見られることは、複雑な書体の影響を示しています。ラテン文字専用のフォントがmarkテーブルを必要とすることは少ないからです。一方、多くのラテンフォントでパンアフリカンサポートを増やすGoogleの取り組みも、markテーブルの存在を増加させている可能性があります。

- **スタイリスティックセット（`ss01`、`ss02` など）と代替**: これらはデータに健全ですが低いレベルで表示されます。約12%のフォントにss01があり、より小さな割合でss02、ss03など、またはsalt（スタイリスティック代替）機能があります。これは特定の文字のオプションの代替デザイン（異なる「a」や「g」など）を提供するフォントに対応しています。

- **大文字とスモールキャップス**: smcp（スモールキャップス）やc2sc（大文字からスモールキャップスへ）などの機能は一部のフォント（数パーセント）に存在し、より高度な活版整理ファミリーを反映しています。

- **書体固有の機能**: 例えば、rlig（アラビア語で使用される必須合字）や様々なインド系機能（上基底、下基底置換のabvs、blwsなど）はそれぞれ約1%以下のフォントに表示されます。これは上で議論した書体のフォントのシェアとほぼ一致しています。

総合的に、これはウェブフォントがより洗練されてきていることを意味します。今年のウェブクロールで見られたフォントの半数以上が少なくとも1つのOpenTypeの手を持っており、多くが12以上を持っています。合字やカーニングのような一般的なものは今では期待されており、単なる特別な特典ではありません。

では、開発者はCSS でこれらの機能を使用しているでしょうか？多くのブラウザエンジンはデフォルトで `liga` や `kern` などのコア機能を適用します（無効にされない限り）。したがって、一部の機能は作者が何もしなくても使用されます。しかし、作者はCSS `font-feature-settings` またはより高度な `font-variant` プロパティを通じて機能を明示的に制御できます。

低レベルの `font-feature-settings` の使用は絶対値では比較的小さいです。このプロパティを通じて特定のOpenType機能を手動で設定するページは約2〜3%に過ぎません。CSSで最もよく切り替えられる機能は `kern` と `liga` であり、一部のCSSリセットやフレームワークがカーニングをオンにしたり合字を意図的にオン/オフしたりするためと考えられます。2025年には、 `font-feature-settings: kern` はデスクトップページの約2.7%に表示され、 `liga` は約2.5%に表示されます。その後、ドロップがあります。 `tnum` （表形式数字）は〜0.8〜0.9%、 `palt` （一部のCJKスペーシング調整に使用される比例代替幅）は〜0.7〜0.8%、そして `lnum` （ライニング数字）や `pnum` のようなものは〜0.4%です。スタイリスティックセット（ `ss01` など）や任意合字は直接設定で〜0.1%以下のページにのみ表示されます。

ただし、任意の `font-feature-settings` 使用（ `normal` を設定するだけかライブラリ経由で含むものも含めて）があるページを数えると、その数は高くなります。約11〜12%のページに少なくとも1つの `font-feature-settings` 宣言があります。これは多くのサイトがフォント機能に触れるボイラープレートやフレームワークを含んでいることを示唆しています（例えば、特定のケースで合字を無効にするために `font-feature-settings: "liga" 0` を設定したり、一部の古いキットが特定の機能を有効にしたりするなど）。

対照的に、より人間に優しい `font-variant` CSSプロパティは約5〜6%のページで使用され

ています。最も一般的なのは `font-variant-numeric` で、表形式数字や旧スタイル数字をリクエストするためによく使用されます。例えば、`font-variant-numeric: tabular-nums` は3.2〜3.3%のページで見られ、`font-variant-caps: small-caps`（〜1.5%のページ）などのプロパティの小さな使用が見られます。少なくない数のページ（約0.5%）が合字を意図的に無効にするために `font-variant-ligatures: none` を使用しています（CSSリセットの一部として、またはモノスペースコードで合字が一般的に不要な場合などによく見られます）。

ここでのトレンドは、開発者がOpenTypeコントロールを徐々に採用していますが、普遍的ではないということです。ほとんどはブラウザのデフォルト（通常は基本を処理する）に依存しています。約10ページに1つのサブセットが、フレームワーク経由または手動で何らかの高度な調整を含んでいます。最も頻繁な意図的な使用は、数字の整列（表形式対比例）、合字やカーニングの明示的な有効化/無効化、そして場合によってはスモールキャップスの使用です。直接の `font-feature-settings` の使用が `font-variant` の使用の約2倍であるという事実は、フレームワークが低レベルの構文を出力していることを示している可能性があります。または、特定の高レベルプロパティが存在しないため、スタイリスティックセットのような特定の一般的でない機能について、開発者が `font-feature-settings` を使用するしかない場合もあります。

日常的なガイダンスとして：フォントが合字などの機能を持っている場合、ブラウザのデフォルトで既に機能している可能性が高いという良いニュースがあります。また、特別な機能（例えば表形式数字）が必要な場合、小さくても増加中のサイトがCSS経由でそれらを使用する方法を示しています。より多くのデザイナーが利用可能な細かいタイポグラフィコントロールを認識するにつれ、これらの数字は引き続き上昇すると予想されます。

可変フォント

可変フォント（1つのファイルでウェイトや幅などの複数のスタイルを許可するもの）は、ここ数年で実験的なものからかなりメインストリームへと移行しました。2025年のデータは採用の継続的な成長を示しています。

可変フォントはどれほど人気があるでしょうか？今年は、39.4%のデスクトップウェブサイトと41.3%のモバイルウェブサイトが自分のページに少なくとも1つの可変フォントを使用しています。つまり、今では10のサイトのうち約4つが可変フォントを使用しています。これは2024年のデスクトップで約33%、モバイルで34%から増加しています。成長（年間〜6〜7パーセントポイント）は安定的で重要です。注目すべきは、モバイルが一貫して可変フォントの採用においてデスクトップを小さなマージンでリードしていることです。これはモバイルのパフォーマンスがフォントファイルの組み合わせからより多く恩恵を受けるか、多くの現代のモバイルファーストサイトまたはPWAがその柔軟性のために可変フォントを採用しているためかもしれません。

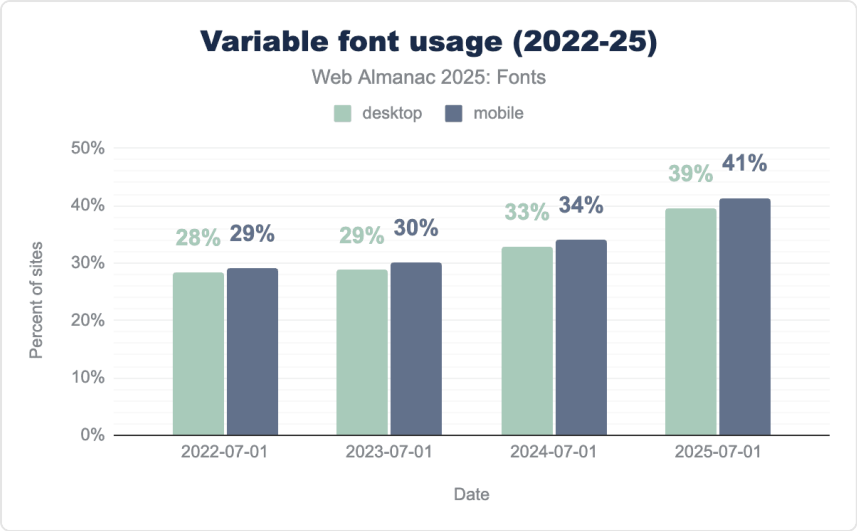


図1.26. デバイス別の可変フォントの使用状況。

ただし、特定の可変フォントの使用分布は非常に上位集中しています。少数のフォントファミリーが非常に大きなシェアを占めています。ウェブで最も使用されている単一の可変フォントは依然としてNoto Sans JP（GoogleのNotoファミリーの日本語サンセリフ）です。2024年には、Noto Sans JPは可変フォントを使用するすべてのサイトの約27%で見つかり、2025年には可変フォントを使用するページの約25〜26%に表示されています。これによりNoto Sans JPはリーチ数でトップの可変フォントとなり、東アジアおよびCJKコンテンツのために世界中で多くのサイトがこのフォントに依存していることを証明しています。その後、次の3つの最大の可変フォントはRoboto、Open Sans、Montserratです。Noto Sans JPと合わせると、これら4つのファミリーはデスクトップとモバイルの両方で全可変フォントリクエストのほぼ60%を占めています。具体的には、Robotoは可変フォントサイトの約15%、Open Sansは11%、Montserratは7〜8%で使用されています。これらはすべてGoogle Fontsで可変形式で利用可能であり、Googleがこれらをデフォルトで可変フォントとして提供し始めたことがその優位性を説明しています。

トップ4を超えると、他の注目すべき可変フォントのクラスターが見られます。Noto Sans KR（可変フォントサイトの〜5%、韓国語の使用を反映）、Noto Serif JP（〜4%）、Noto Sans TC（〜2%）で、これらはすべてCJKスクリプトの同じNotoファミリーの一部です。次に、Inter（〜3.3%）、Raleway、Oswald、Playfair Display、Rubik、Roboto Slab、Google Sans、Nunitoなどの人気のラテン可変フォントがそれぞれ可変フォントを使用するサイトの2〜3%に登場します。このロングテールは、多くの可変フォントが存在する一方で、スケールで使用されているものはほとんどが大きな問題（複数のスクリプトや複数のスタイルを効率的にカバーするなど）を解決し、入手が容易なもの（主にGoogle経由のオープンソース）であることを示しています。

これらの可変フォントがどのように使用されているかを観察することも興味深いです。可変フォントでCSSに設定される一般的な軸は洞察を提供します。ウェイト軸（`wght`）は断然最も使用されており、可変フォントを使用するページの約57〜61%でウェイトが明示的に調整されています。これは、ほとんどの人が可変フォントを基本的に1つのファイルに複数のウェイトを持つために使用していることを示しています（例えば、異なるブレークポイントでテキストを太くしたり細くしたりする）。ただし、次に一般的な軸は特定の可変フォントシステムから来ています。`FILL`と`opsz`（光学サイズ）はそれぞれ可変フォントページの約35〜39%に表示され、`GRAD`は約27%で続きます。これらはMaterial Symbols アイコンフォント（グリフフィル、グレード、光学サイズの軸を持つ）に対応しており、Googleが広く使用している可変アイコンフォントです。したがって、可変フォント使用の大部分は実際にはテキストではなく、複数のスタイリスティック軸を持つアイコンによるものです。幅（`wdth`）とスラント（`slnt`）が使用中の他の主要な軸です。スラントはデスクトップの約19%（モバイルの24%）の可変フォントページで見られ、幅は両方の約17%で見られます。これは、より小さいが確固たる割合のサイトが幅を調整するか、中間的なスラントを使用している（おそらく真のイタリックの代替またはレスポンシブ調整として）ことを示しています。他のすべての軸（カスタムデザイン軸やRoboto Flexのようなより細かい軸など）は使用率が5%以下または1%以下で表示されており、デモや非常に特定のプロジェクトからのノイズである可能性もあります。

実際には、ウェブの可変フォントは主にウェイト調整のために使用されており、場合によっては幅やスラントにも使用されています。`opsz`、`FILL`、`GRAD`の高い表示は、広いトレンドではなく1つの人気アイコンフォントの使用の産物のようです。可変フォントの約束（継続的なバリエーションによる細かいタイポグラフィの調整）は、高いレベルの技術的統合によってサポートされていますが、ほとんどのデザイナーにとってクリエイティブな探索の初期段階にあります。多くの人が可変フォントを便利なマルチウェイトファイルとして使用しており、まだタイポグラフィ表現のための完全に動的なリソースとしてではありません。

そして、可変フォントアニメーション（フォントバリエーションプロパティでCSSトランジションまたはキーフレームを使用）は非常にニッチです。2025年の全ページの約0.3%のみが、可変フォント軸を含むアニメーションまたはトランジション（低レベルの`font-variation-settings`経由、または可変フォントの`font-weight`やその他の短縮形経由）を含んでいます。可変フォントを使用するページの中でも、何らかのアニメーションを行うのは1%未満（約0.7〜0.8%）に過ぎません。それが起こる場合は通常は装飾的で、例えばヒーロータイトルの拡大・縮小、またはウェイトを変更するホバーエフェクトを持つアイコンボタンなどです。レイアウトシフトを引き起こす可能性があるため、または単にそのデザイン空間に踏み込んでいる人が多くないため、一般的ではないようです。

カラーフォント

カラーフォント（絵文字やアイコンによく使われる多色グリフを持つフォントを提供する新

技術）は興味深い技術でしたが、ウェブでの採用は依然として非常に限られています。2025年のデータはほぼゼロの基盤からの若干の増加を示していますが、まだメインストリームからは程遠いです。

0.06%

図1.27. カラーフォントを使用しているモバイルページの割合。

クロールで観察されたウェブサイトの約0.05〜0.06%のみがカラーフォントを使用していました。これは2,000ページに1つのオーダーです。数年前の約0.01%から成長していますが、依然として5倍の増加であり、全体的には微小です。生の数字で言えば、約6,000〜7,000のデスクトップページと同様の数のモバイルページでカラーフォントを見つけました。使用量は「倍増」していますが、このレポートの他の数字と比較すると、依然として四捨五入の誤差の領域に居ます。

なぜこれほど少ないのでしょうか？いくつかの理由があります。最近まで、新しいカラーフォントフォーマット（COLR/CPAL v1）のクロスブラウザサポートが不完全でした。カラーフォントを作成したり、絵文字以外のユースケースを見つけることは一般的ではありませんでした。そして多くの潜在的なユース（アイコン、シンボル）は代わりにSVGや画像として提供されていました。

カラーフォントを使用している少数のサイトを見ると、何のために使用されているのでしょうか？データは、カラーフォントを持つページの大多数が絵文字や特殊なアイコノグラフィのために使用していることを示しています。実際、断然最も一般的なカラーフォントはNoto Color Emoji（COLRとCBDTビットマップを含む複数のフォーマットを持つGoogle/Androidの絵文字フォント）です。Noto Color Emojiはカラーフォントを使用するデスクトップページの約73%（およびモバイルのカラーフォントページの多数）に表示されています。基本的に、カラーフォントを使用しているほとんどのサイトは、OSの絵文字に頼るのではなく、一貫した外観で絵文字をレンダリングするためにそれを使用しています。その後、かなりの部分は2つの日本語カラーSVGフォント（特定のサイトで装飾的なテキストとして使用される）によるもので、それぞれカラーフォント使用ページの7%と5%を占めています。それ以外はロングテールです。Twemoji（Twitterの絵文字）が数パーセント、カラーレイヤーを持ついくつかのアイコンフォント、そして一握りの装飾的な多色テキストフォントです。

カラーフォント技術の観点から、いくつかのフォーマットがあります。SVG-in-OpenType、COLR/CPAL（v0とv1）、および古いビットマップフォント（Googleのビットマップの CBDT/CBLC、AppleのSbix）。2025年のフォントリクエスト別の使用内訳は、大まかにSVGが58〜60%、COLR v0が25%、COLR v1が16%、CBDT/sbix（ビットマップ）が数パーセントです。SVGが最も高いのは興味深いことです。これは実際にはSVGベースで高トラフィックページで使用されているあの2つの日本語フォントのためです。COLR v0（レイヤードベ

クターグリフフォーマットの最初のバージョン）は多くの現在のカラーフォント（一部の絵文字セットやアイコンフォントを含む）で使用されています。グラデーションを許可し絵文字にとってよりコンパクトなCOLR v1は新しいですが、すでに〜16%を占めています。これはNoto Emoji COLR v1のような更新された絵文字フォントから来ている可能性が高いです。ビットマップフォーマット（GoogleのカラーフォントのCBDT、AppleのSbix）は今やウェブ上ではほぼ無視できるほどです。それらはネイティブアプリ向けでした。

もう1つの注目すべき現象は、一部のカラーフォントが互換性のために複数のフォーマットを含んでいることです。例えば、Noto Color EmojiはCBDTとCOLRの両方のテーブルを持っている可能性があります。これは「ファミリー」でカウントすると、そのフォントを複数のフォーマットカテゴリでダブルカウントする可能性があることを意味します。しかし、使用の観点からは、ブラウザサポートの増加でCOLR v1が勢いを得ています（絵文字にとってSVGやビットマップよりもはるかに効率的です）。

カラーフォント内のカラーパレット（フォントが異なるプリセットのカラースキームを提供するもの）は広く使用されていません。カラーフォントの95%以上は、定義されたパレットがゼロまたは1つです。つまり、固定された色のセットを持っているか、デフォルトに依存しています。複数のパレットを持つのはごくわずかです（例えば、パレットメカニズムを通じてスキントーンやテーマを切り替えられる絵文字フォントなど）。そして複数のパレットを持つカラーフォントは、多くの場合せいぜい2〜3つです。今日のほとんどのカラーフォントはテーマカラーの切り替えを提供することを目指していません。必要な色をハードコードしているだけです。これは当然のことです。テーマの切り替えの場所はCSSでのフォントパレットオーバーライドにあり、フォントにあるのではありません。

今日使用されているほとんどのカラーフォントのカラーレイヤーの数も小さいです。多くのアイコンフォントは2色のみ（例：前景と背景）を使用する可能性があります。より多くのレイヤーはフルの絵文字セット（すべての絵文字文字を構築するために数百のレイヤーを持つ可能性がある）にのみ表示されます。

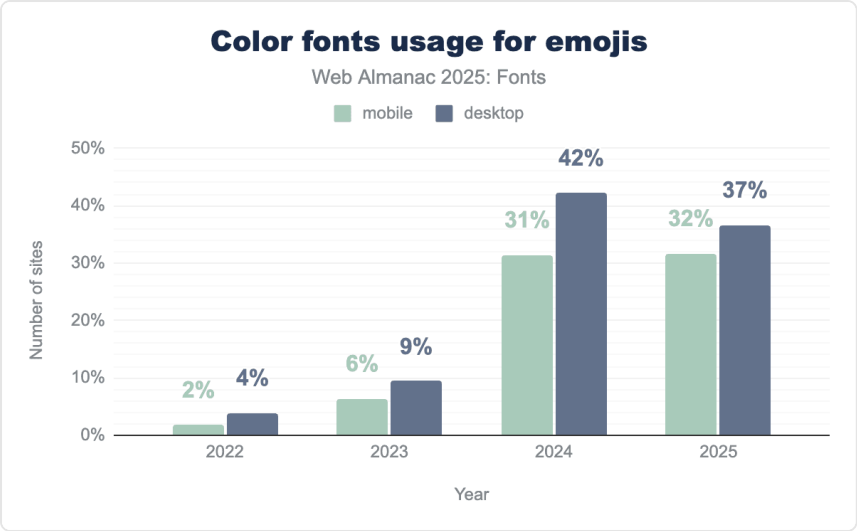


図1.28. 絵文字用のカラーフォントの使用状況。

量的に見ると、カラーフォントリクエストの約32〜37%が絵文字用であることがわかります（このパーセンテージは2024年に上昇し、他の用途が成長するにつれて今年はわずかに低下しています）。そのトレンドは、カラーフォントリクエストの多数（〜63〜68%）が現在は非絵文字用途（装飾的なCJKフォントや多色アイコンなど）であることを意味します。しかし、ページ数では、絵文字フォント（特にNoto Color Emoji）は依然としてカラーフォントの主要な使用です。一部の大きなプラットフォームやサイトがNoto Color Emojiをフォールバックとして含み、多くのページに貢献している可能性がある一方で、他のカラーフォントは少数のページで使用されているが、多くのページビューを持つサイトの装飾フォントのように多数回フェッチされる可能性があります。

2025年、カラーフォントはウェブ上の新奇なものとして残っています。成長していますが、微小な基盤から始まっています。ほとんどの開発者はそれらを必要としないか、過去のサポートの問題のために避けています。カラーフォントを使用している人は、主に一貫した絵文字レンダリングを確保するか、非常にカスタムな何か（多色のロゴタイプやアイコンなど）を行っています。ブラウザサポートが安定するにつれ（COLR v1は現在Chrome、Edge、Firefoxでサポートされています）、より多くの使用が見られるかもしれません。例えば、一部の人々がOS間で一致するカラーの絵文字を持つために絵文字フォントを埋め込み始める可能性があります。しかし今のところ、高度なフォーマットの観点から、カラーフォントはウェブ上の可変フォントより数百倍一般的でありません。

結論

2025年のウェブフォントの状況は、主要な変化ではなく段階的な改善が続く、成熟しほぼ遍在する技術を描いています。過去10年間で、ウェブフォントは周辺的な機能からウェブの本質的な部分へと変貌しました。10のサイトのうちほぼ9つが使用しており、その割合は毎年ゆっくりと完全な飽和に向かっています。

過去数年に観察されたいくつかの主要なトレンドが確固たるものとなりました：

- **セルフホストとサービスの比較:** ウェブフォントの配信はサービスからセルフホストへとゆっくりと移行しています。Google Fontsは依然として全ウェブサイトの半数以上にフォントを提供していますが、パフォーマンスやプライバシーのために自分でフォントをホストすることを選ぶ開発者が増えるにつれ、そのシェアは緩やかに低下しています。今では約3分の1のサイトがすべてのフォントをセルフホストしており、数年前の約5分の1から増加しています。別の大きなシェアのサイトはGoogleのCDNとセルフホストのハイブリッドを使用しています。これは、開発者がサードパーティのインフラに完全に依存しているわけではなく、キャッシングとローディングを自分のニーズに合わせて最適化できるバランスを示しています。

- **フォーマットの統合:** ウェブはWOFF2をプライマリフォントフォーマットとして標準化しました。フォントファイルの約65%はWOFF2であり、WOFFと合わせると約81%です。生のTTF、EOT、SVGフォントなどの古いフォーマットはほぼ完全に廃止されています。この統合はパフォーマンスに優れており（WOFF2は圧縮が優れています）、開発者がサポートする必要があるものを簡素化します。データは、ほとんどのサイトがこの方向に従っていることを示していますが、一部のセルフホスト環境はまだWOFF2の提供または正しいMIMEタイプの送信で遅れています。

- **フォントのサイズとパフォーマンス:** 典型的なウェブフォントは30〜40 KBの範囲（圧縮後）であり、サイトはしばしば複数のフォントを使用するため、フォントは依然としてわずかな重量コストを持っています。中央値のサイズは最近わずかに増加しており、より豊富なフォント（例：可変フォントやより多くのグリフを持つフォント）を反映しています。フォントの相当部分は100 KBを超えており、特にアジア系書体やサブセット化されていない場合です。これはサブセット化と慎重なローディング戦略の重要性を強調しています。これらのフォントのほとんどが現在WOFF2であることはコストを軽減するのに役立ちますが、開発者

は依然として注意が必要です。多くの大きなフォントファイルの多用はロードスピードを損なう可能性があります。2025年のクロールデータは、ほとんどのサイトがフォントの使用において適度であることを示していますが、小さなパーセンテージがやり過ぎで配布の「テール」を劇的に歪めています。

- **グローバルスクリプト:** ウェブタイポグラフィはよりグローバルにフォーカスされるようになり、ラテン文字を超えた言語のサポートが大きくなっています。毎年、キリル文字、ギリシャ語、アラビア語、ヘブライ語、デバナーガリー、CJKなどがウェブフォントの使用に意味のある存在として見られます。これは主に、これらの書体の高品質フォントを生産する協働プロジェクトとオープンソースの取り組みのおかげです。ウェブフォントがほぼ独占的にラテン文字向けに開発されていた2010年代初頭からの大きな変化です。今では、開発者はさまざまな書記体系のウェブフォントを見つけることを合理的に期待できます。データは特に、Notoなどのフォントファミリーのおかげで成長を続ける日本語と韓国語のウェブフォントの採用を強調しています。中国語はファイルサイズのために依然としてより困難な問題ですが、インクリメンタルフォント転送（IFT）—フォントの必要な部分だけをオンザフライでロードする方法—などの技術が将来それを変える可能性があります（これは地平線上にあり、巨大なフォントにとってのブレークスルーになる可能性がありますが、依然として実験的です）。

- **可変フォント:** かつては最先端のコンセプトだった可変フォントは、今やメインストリームです。約40%のウェブサイトがそれらを使用しており、しばしばGoogle Fontsが複数の静的ファイルの代わりに可変ファイルを提供することで透過的に行われています。豪華なアニメーションやコンテンツの継続的な変動性という点でデザインをまだ革命的には変えていませんが、実用的な利益をもたらしました：フォントリクエストを簡素化し、複数のスタイルが必要な場合に総バイト数を削減することがあります。多くの人気フォントは現在デフォルトで可変です（Robotoなど）、そして開発者はほとんどの場合、以前に複数のウェイトを使用してきたように使用しています—ただし、希望すればより細かい制御のボーナスがあります。

- **カラーフォント:** カラーフォント（COLR/CPAL、SVGグリフ）のサポートが成長しているにもかかわらず、それらはウェブ上でまだ稀です。絵文字の使用を除いて、多色のグリフを利用しているサイトは少ないです。注目すべき領域として残っていますが、2025年現在、カラーフォントはまだ一般的なウェブ使用において足場を見つけていません。

- **レンダリングのためのCSS:** より多くのサイトが利用可能なCSSを使用してフォントのロードとレンダリングを最適化しています。`font-display`は現在フォントを使用するサイトの多数に指定されており、FOIT/FOUTについての意識的な選択を反映しています。`swap`の主要な使用は、アイコンフォントのみが整合性のために意図的に遅延させていて、テキストの素早い表示に対する共有の優先度を示しています。`preconnect`や`preload`などのリソースヒントはユニバーサルではありませんが、健全な少数派がこれらの技術を採用しており、パフォーマンスを意識した開発プラクティスが増加していることを示しています。また、まだニッチですが、`text-wrap: balance`や`hyphens: auto`などのプロパティの採用は、より細かいタイポグラフィの調整がテキストレイアウトの改善のために徐々に普及していることを示しています。本質的に、ウェブ開発者はすべてをデフォルトに任せるのではなく、フォントのロード方法とテキストのフロー方法をより積極的に管理しています。これはエコシステムが成熟しているサインであり、フロントエンド開発者が他のパフォーマンスやUXの側面と同じ真剣さでタイポグラフィを扱っています。

全体的に、2025年のウェブフォントは収束と洗練の物語を語っています。フォーマット（あらゆる場所でWOFF2）、配信パターン（セルフホストが増加しているが、サービスも依然として重要）、広く使用されるファミリー（少数のウェブフォントが今や事実上コアになっている）、ベストプラクティス（`font-display: swap`、重要なフォントのプリロードなど）の収束が見られます。洗練にはグローバル言語サポートの改善、OpenType機能の使用、テキスト表示の粗い端を滑らかにするための新しいCSS機能が含まれます。したがって、ウェブフォントの景観は概して安定していますが、停滞していません。単一の変化がヘッドラインを掴むことはありませんが、これらの発展が集合的にウェブをより磨かれたものにしてアクセス可能にしています。

フォントの次の主要な発展は何でしょうか？述べたように、インクリメンタルフォント転送（IFT）は非常に有望な新技術です。ファイル全体をフロントローディングするのではなく、ブラウザが特定のテキストセットのためにフォントの必要なグリフのみをダウンロードすることを可能にします。標準化され採用されれば、IFTはCJK、絵文字、アイコンセットのような巨大なフォントにとって革命的になる可能性があります—「フォントサイズ」の問題を効果的に解決します。この技術はまだ開発中ですが、来年のWeb Almanacにはアイナンスルおよびインパクトに関するデータが報告できるかもしれません。

そして、2025年のチャプターは自信を持って幕を閉じます：フォントはウェブの基本的な構成要素であり、ほとんどの開発者によって賢く使用されており、前途はそれらをより速く、より包括的に、より表現豊かに、そして誰にとっても扱いやすくすることです。

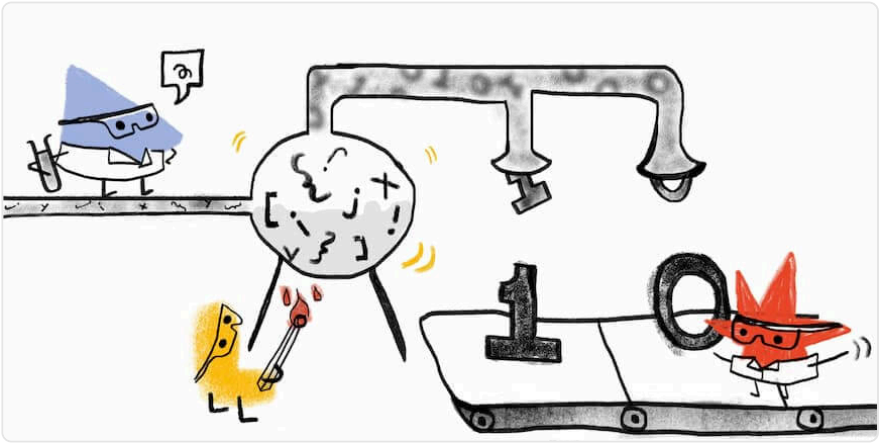
著者

Charles Berret

charlesberret https://charlesberret.net/

Charles Berretはジャーナリスト、開発者、メディア研究者であり、情報技術の歴史と哲学を研究しています。ニューヨーク市在住で、Enigmaにてデータからストーリーを見つける仕事をしています。

部 I 章 2
WebAssembly



Nimesh Vadgama - VN によって書かれた。
Nurullah Demir と Barry Pollard によってレビュ。
Nimesh Vadgama - VN による分析。
Barry Pollard 編集。
Sakae Kotaro によって翻訳された。

はじめに

WebAssembly（Wasm）は、ウェブ中心の最適化ツールから高性能なユニバーサルバイトコードフォーマットへと進化しました。2019年にW3C標準として正式に採用され[1]、2025年12月のWasm Version 3.0のリリース[2]でエコシステムは技術的な転換点を迎えました。このバージョンはガベージコレクション、64ビットアドレス空間、Multiple Memoriesなどの高度な機能[3]を標準化しており、Java、Kotlin、Dartなどのハイレベルな Languageがブラウザとスタンドアロン環境の両方でネイティブかつ効率的に動作することを可能にしています。

1. *https://www.w3.org/TR/2019/REC-wasm-core-1-20191205/*
2. *https://webassembly.github.io/spec/core/*
3. *https://webassembly.github.io/spec/core/*

方法論

WebAssemblyが初めて紹介された2021年版Web Almanac[4]と同じ方法論に従っています。

データ収集: 本章は、Google BigQueryにホストされているHTTP Archiveの2025年7月クロールデータを利用して、`Content-Type`（`application/wasm`）と`.wasm`ファイル拡張子を照合することでWebAssemblyモジュールを特定しています。この方法で、43,000サイトで少なくとも1つのWebAssemblyモジュールを発見し、分析した全サイトの0.35%を占めています。

分析: HTTP Archiveデータセットに加えて、HTTP Archiveから特定されたWebAssemblyモジュールをダウンロードして検証するツール`almanac-wasm-stats`をローカル分析に使用しています。このツールはダウンロードしたファイルからメタデータを抽出し、Wasmモジュール内で使用されているプログラミング言語、ライブラリ、特定の機能を特定することを可能にします。

制限事項: `almanac-wasm-stats`ツールはWasmモジュールの静的解析に焦点を当てており、実行はしません。そのため、ブラウザやスタンドアロン環境での実際の実行中に存在する可能性のある動的な動作やランタイム機能を捉えることができません。また、一部のWasmモジュールは難読化またはminify化されている場合があり、その特性を正確に特定する能力が制限される可能性があります。 言語使用分析に役立つ以下の機能を(wasm-stats[5])を拡張してalmanac-wasm-statsに実装しました。

1. URLと対応するユーザーエージェント文字列を入力として受け取ることで、ダウンロードタスクを改善します。
2. BigQueryのJSONL結果形式で大量の入力を受け付けます。
3. Binary ToolkitでWasmを検証し、統計改善に役立つインサイトを提供します（wasm2wat参照）。
4. Wasmファイルのダウンロード、検証、統計収集などのタスクを並行して実行・追跡します。
5. 古いRust実装（`wasm-stats`）の言語識別子を拡張し、Scala、Dotnet/Mono、Go & TinyGo、TeaVMベースの言語、Kotlinを新たに追加。これにより「Unknown」の数が減少し、言語統計が改善されます。
6. 検証とダウンロードの失敗とともに、完全なWasm言語使用統計を生成します。
7. 将来の拡張に向けて、既存のJSON統計形式でWebAssembly Toolkit / SDKを使用した新しい統計を導入できるプラグアンドプレイアーキテクチャを採用しています。

4. *https://almanac.httparchive.org/ja/2021/webassembly#方法論*
5. *https://github.com/HTTPArchive/wasm-stats*

WebAssemblyの使用状況

このセクションでは、ウェブにおけるWebAssemblyの使用状況に関する結果を紹介します。

年次トレンド

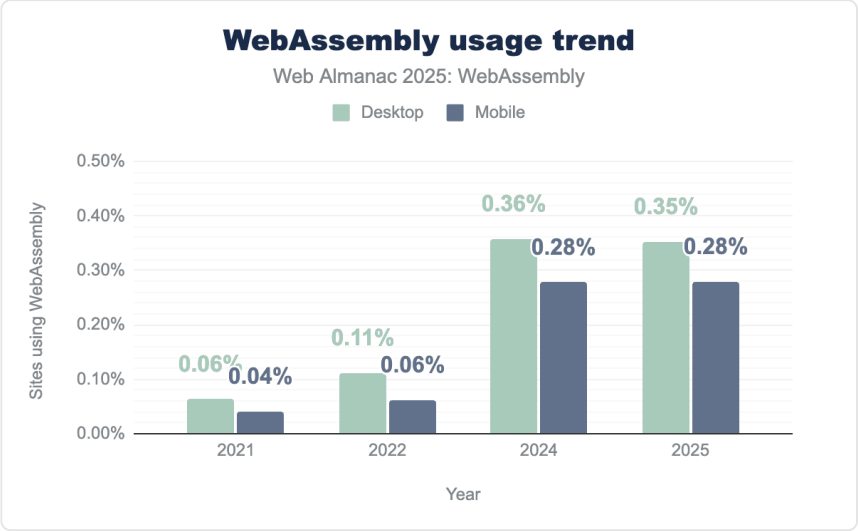


図2.1. WebAssemblyの使用トレンド

WebAssemblyの採用は2021年の0.04%から成長していますが、直近2年間は一定の水準を維持しています。2025年には、デスクトップサイトの0.35%、モバイルサイトの0.28%でWebAssemblyモジュールを確認し、約43,000サイトに相当します。また、観測された全ての年でモバイルの採用率はデスクトップより低く、平均30%の差があります。

ランク別WebAssembly

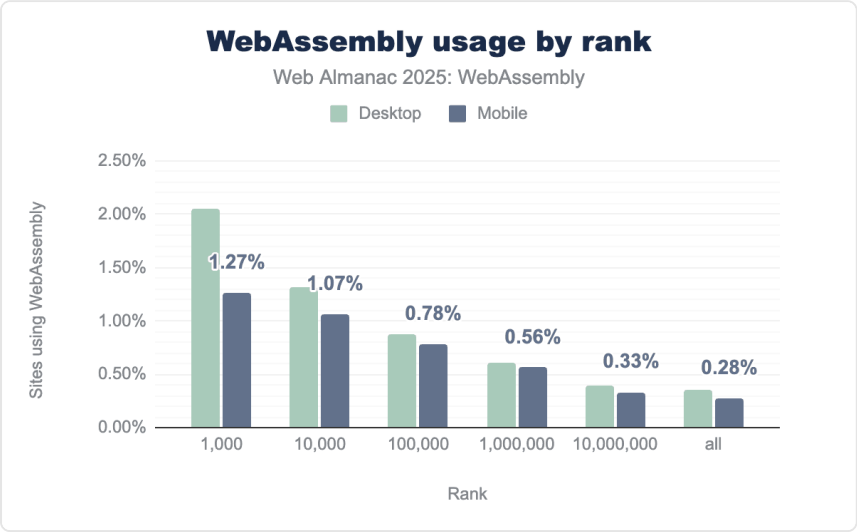


図2.2. WebAssembly使用サイトのランク

サイトの人気度とWebAssemblyの採用には強い相関関係があります。使用はトップ1,000サイトに最も集中しており、デスクトップで2%、モバイルで1.27%に達しています。これらのトップクラスのサイトは、Wasmが提供する高いパフォーマンスを必要とするデザインツールや重いメディアエディターなどの複雑なアプリケーションを頻繁にホストしています。

サイトのランクが下がるにつれて採用率も低下し、一貫した分布パターンに従っています。トップ1,000万サイト外では、採用率はデスクトップで約0.33%、モバイルで0.28%です。

トップランクグループではデスクトップの使用率が高いままですが、ロングテールではその差が大幅に縮まっており、ウェブの大多数においてWebAssemblyは特定の環境に限定されるのではなく、クロスプラットフォームリソースとして展開されていることを示しています。

WebAssemblyのリクエスト

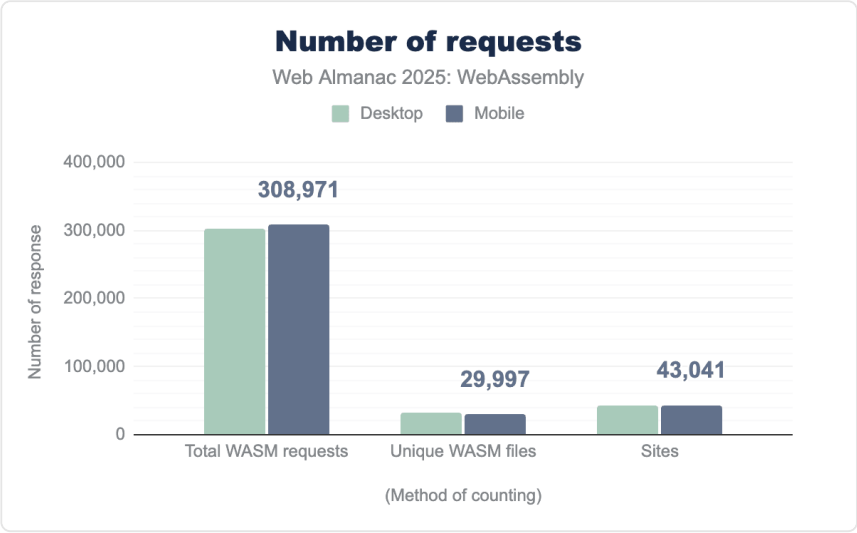


図2.3. Wasmリクエスト数

全体で、デスクトップで303,496件、モバイルで308,971件のWebAssemblyリクエストを記録しました。デスクトップサイトの方がWebAssemblyを多く利用していますが、リクエストの総量はモバイルがわずかに多くなっています。

さらに、デスクトップで157,967個、モバイルで165,870個のユニークなURLを特定しました。ユニークなバイナリ数を推定するために、同一のファイル名とレスポンスサイズでモジュールをグループ化しました。この方法で、デスクトップで87,596個、モバイルで84,851個のユニークなWasmモジュールが見つかりました。これらの結果は、名前ベースで約72%のWebAssemblyリクエストが重複モジュールを提供していることを示しており、ウェブ全体でのライブラリの大規模な再利用を浮き彫りにしています。

MIMEタイプ

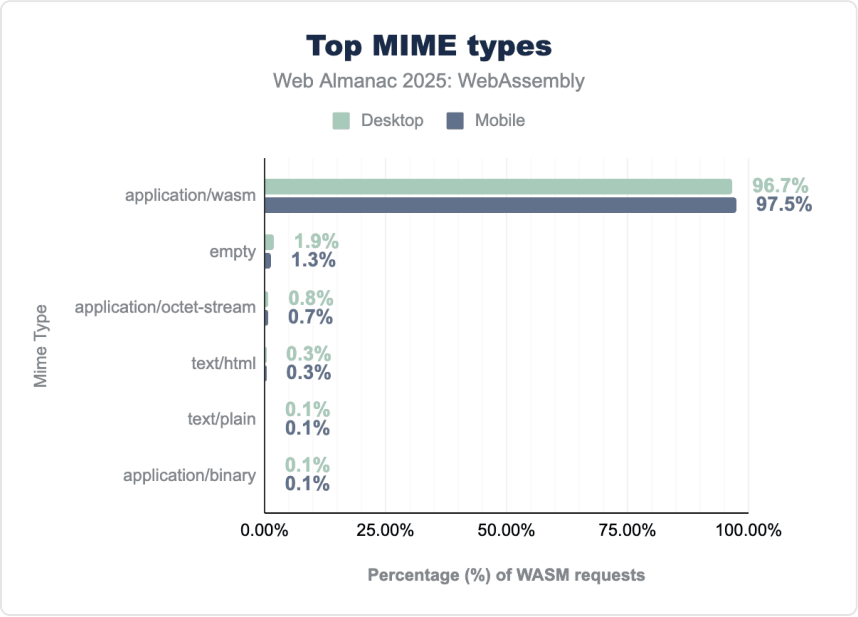


図2.4. 上位MIMEタイプ

`application/wasm` MIMEタイプはデスクトップで293,470件、モバイルで301,127件のリクエストで確認されました。MIMEタイプが欠落または不正確（`text/html`や`text/plain`など）なケースは少なく、デスクトップで3.2%、モバイルで2.4%のリクエストに影響していました。これらは2021年と比較して大幅に減少しており、適切なサーバー設定への意識と遵守が向上していることを示しています。

モジュールサイズ

WebAssemblyモジュールのサイズは使用ケースによって大きく異なります。下位50%のモジュールは2KBから14KBの範囲で非常に小さいことが確認されました。これらは通常、JavaScriptが精度に欠けるパフォーマンスクリティカルなタスクを処理するために、AssemblyScriptやRustで書かれたBase64エンコーダーやチェックサム計算機などの「マイクロユーティリティ」です。

一方、90パーセンタイルではサイズがデスクトップで381KB、モバイルで316KBへと大幅に増加します。これらの大きなバイナリは通常、複雑な3Dレンダリングやロジックを処理するためにC++やC#などの重い言語からコンパイルされた、Adobe PhotoshopやGoogle Earthなどのフルデスクトップグレードのアプリケーションをウェブに移植したものです。

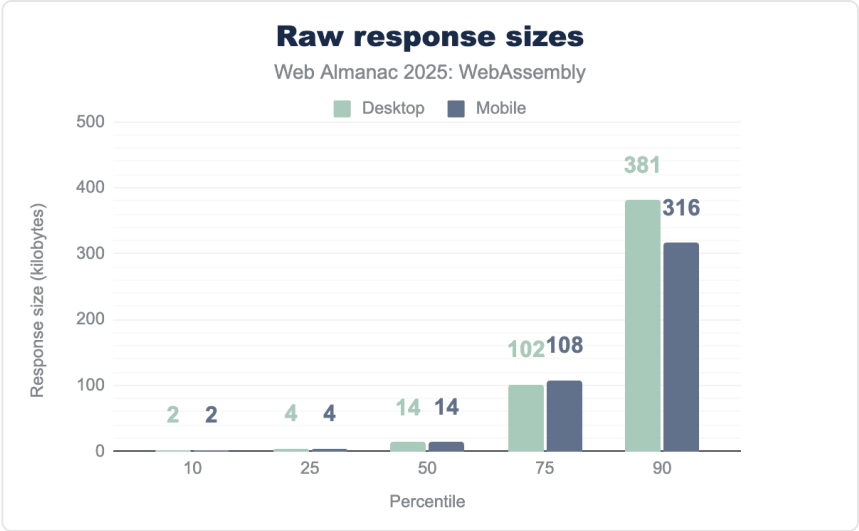


図2.5. 生のレスポンスサイズ。

上記のグラフはレスポンスボディのサイズを示しており、「生のレスポンスサイズ」と呼ばれ、クライアントが受信したデータペイロードのデコード済みの生データのみを測定します。リソース自体のサイズを表しています。ただし、Wasmの成果物に関する研究と一般的な慣行によると、WasmモジュールはBrotliなどの様々なツールで圧縮・最適化され、Content-Encodingヘッダーとともにgzip、br、zstdなどの圧縮方法でネットワーク経由でクライアントに転送されます。

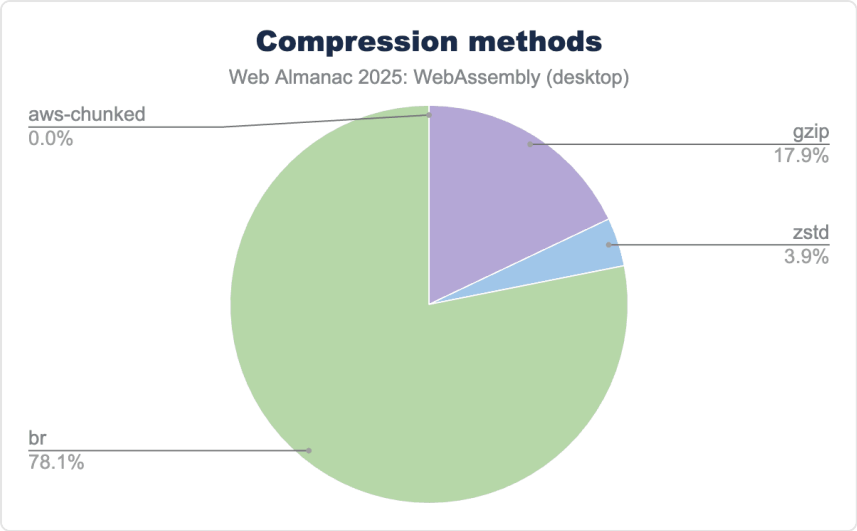


図2.6. デスクトップクライアントで使用される圧縮方法

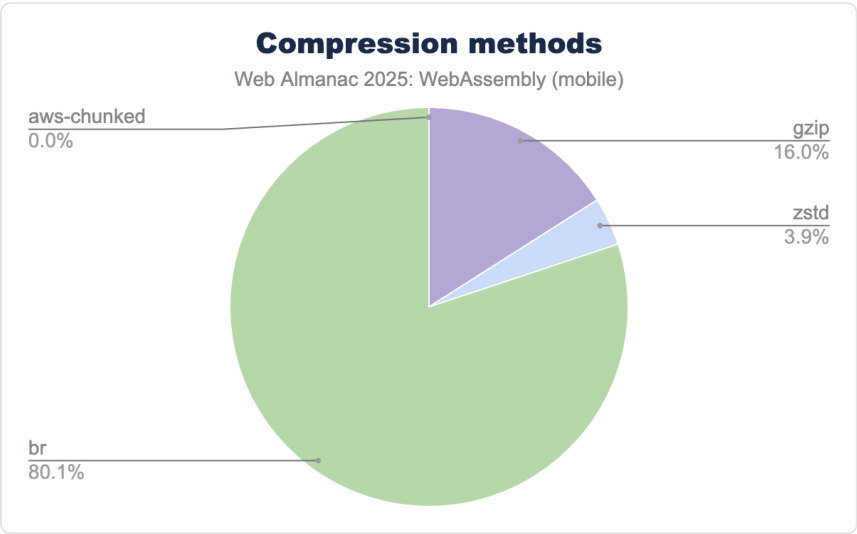


図2.7. モバイルクライアントで使用される圧縮方法

興味深いことに、過去数年間の様々なコミュニティによるパフォーマンスベンチマーク[6]活動を見ると、'br'と'zstd'の圧縮方法への認知が高まっており、開発者やCDNプロバイダーによる継続的な進化と採用が見られます。

6. *https://facebook.github.io/zstd/#benchmarks*

図2.8. 検出された最大のWebAssemblyファイル（デスクトップ）。

図2.9. 検出された最大のWebAssemblyファイル（モバイル）。

これらの標準的な分布を超えて、データセットには重要な外れ値が含まれています。特定された最大の単一WebAssemblyモジュールは、デスクトップで234MB、モバイルで170MBを記録しており、大規模なクライアントサイドアプリケーションが展開されていることを示しています。

興味深いことに、JSの成果物がMBサイズであることが多いのに対し、Wasmの成果物がかなり小さいサイズである理由は、JSが人間が読める高レベルのソースコードであるのに対し、バイトコードはマシンに依存しないコードの低レベルな中間表現であるからです。

Google V8のような現代のJSエンジンは、実行プロセスの一部としてJSソースコードを内部的にバイトコードに変換しています。これはバイトコードのJSに比べた小さなサイズでの効率性を示しています。

WebAssemblyライブラリ

次に、エコシステムで最も人気のある基盤となるライブラリとフレームワークを理解するために、WebAssemblyバイナリ内のインポート名を分析します。

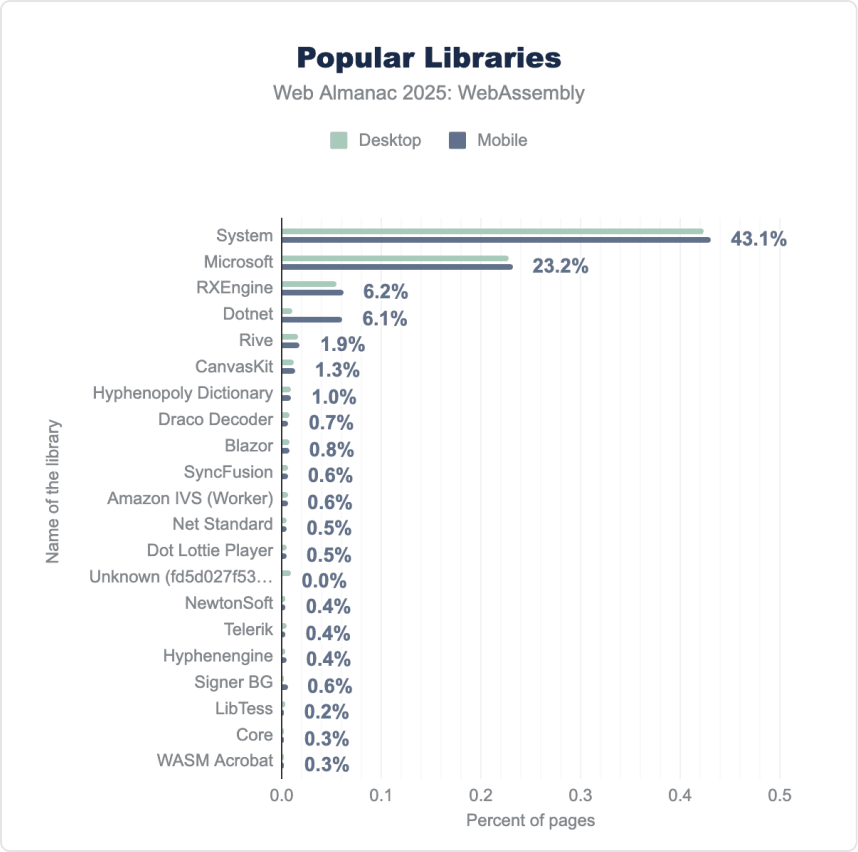


図2.10. 人気のWebAssemblyライブラリ。

WebAssemblyモジュールで最も使用されているライブラリやフレームワークは、System（43%）、Microsoft（23%）、RXEngine（6%）、Dotnet[7]（6%）であり、特にDotnetとBlazor[8]フレームワークによって牽引されるMicrosoftのエコシステムの優位性を示しています。

MicrosoftはSystemユーティリティ、Identity、ネットワーキング、ストレージ、JSONなど、多数の再利用可能なライブラリ向けに様々なWebAssemblyライブラリとフレームワークを持っています。それらのライブラリとフレームワークを組み合わせることで、WebAssemblyにおけるMicrosoftエコシステムはデスクトップで78.8%、モバイルクライアントで79.3%をカバーしています。

7. *https://devblogs.microsoft.com/dotnet/extending-web-assembly-to-the-cloud/*
8. *https://learn.microsoft.com/en-us/aspnet/core/blazor/hosting-models?view=aspnetcore-10.0#blazor-webassembly*

WebAssembly言語

WebAssemblyはC++、C#、Rubyなど様々な言語で開発できます。Wasm 3.0の導入により、Java、Scala、Kotlin、Dartなどのサポート言語の範囲が拡大しました。このセクションでは、WebAssemblyモジュールの開発に使用されている言語の概要を説明します。

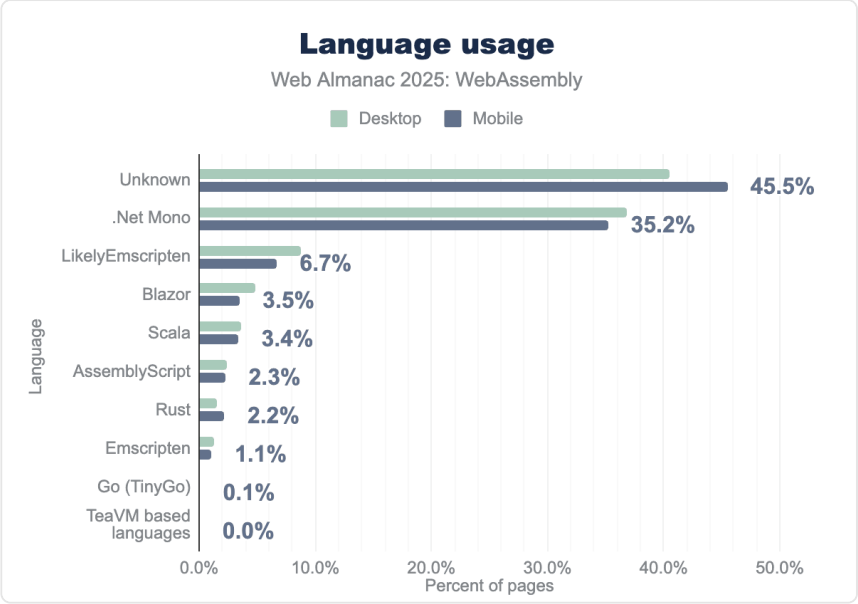


図2.11. WebAssembly言語の使用状況。

ツールはデスクトップで64.3%、モバイルで72.8%のWebAssemblyモジュールのソース言語を正常に特定しました。残りの（35.7%と27.2%）は、主にminify化（メタデータの除去）、WebAssemblyの検証またはダウンロードの失敗により特定できませんでした。

特定された言語の中では、.NET/Mono + Blazorが最も一般的で、デスクトップで41.7%、モバイルで38.7%のシェアを持っています。minify化されたエクスポート/インポート名はソース言語を不明瞭にすることが多いですが、Emscripten（C++）ツールチェーンは独自の命名規則を使用しています。これにより確信を持ってこれらのモジュールを帰属させることができ、Emscriptenがデスクトップで10.1%、モバイルで7.8%を占め、Scalaがデスクトップで3.6%、モバイルで3.4%という結果が得られました。

ライブラリ使用に関する調査結果と合わせると、これらの結果はWebAssemblyの領域におけるMicrosoftエコシステムの圧倒的な優位性を強調しています。

WebAssembly機能

このセクションでは、MVP後（Minimum Viable Product以降）のWebAssembly機能の使用状況を分析します。ここで取り上げる機能はWebAssemblyがサポートする全機能を網羅していないことを認識していますが、これらの機能の採用状況を強調することは重要だと考えています。読者の方々には、将来的により多くの機能を追跡できるよう、本章で使用した分析ツールへの貢献をお願いします。

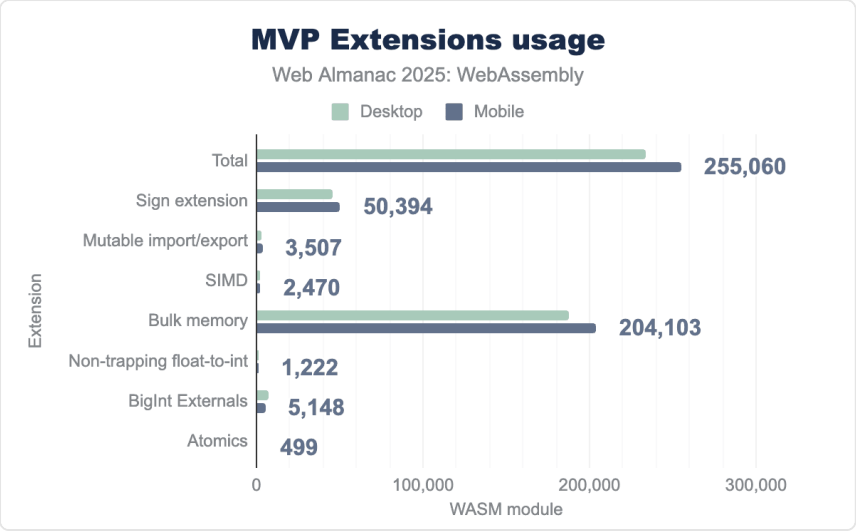


図2.12. MVP後の拡張機能の使用状況。

バルクメモリが最も広く採用されている機能で、デスクトップで187,674、モバイルで204,103のモジュールに登場しています。この機能は大きなメモリブロックの効率的なコピーと初期化を可能にすることでパフォーマンスを向上させ、Cのネイティブな memcpyFunction の効率を模倣しています。さらに、符号拡張（8ビット整数から32ビットへの拡張など、整数値を拡張する演算子を提供する）が2番目に人気の機能で、デスクトップで45,969、モバイルで50,394のモジュールで確認されました。

結論

WebAssemblyの採用は過去4年間で大幅に増加し、2021年の0.04%から2025年の0.35%へと上昇しましたが、直近2年間では成長が安定しています。使用は高ランクのウェブサイトで最も多く、人気の低いページでは大幅に減少します。WebAssemblyは現在、特定のユーテ

9. *https://webassembly.org/features/*

ィリティ機能（暗号化やチェックサムなど）の処理と、完全なスタンドアロンアプリケーションの駆動という2つの異なる目的で展開されています。さらに、調査結果はMicrosoftのフレームワークの広範な採用を示しており、現在のWebAssemblyエコシステムを牽引するうえでの重要な役割を示しています。

Wasm仕様の大きな進展とコミュニティからの関心の高まりを考慮すると、WebAssemblyの採用は今後さらに増加すると考えています。

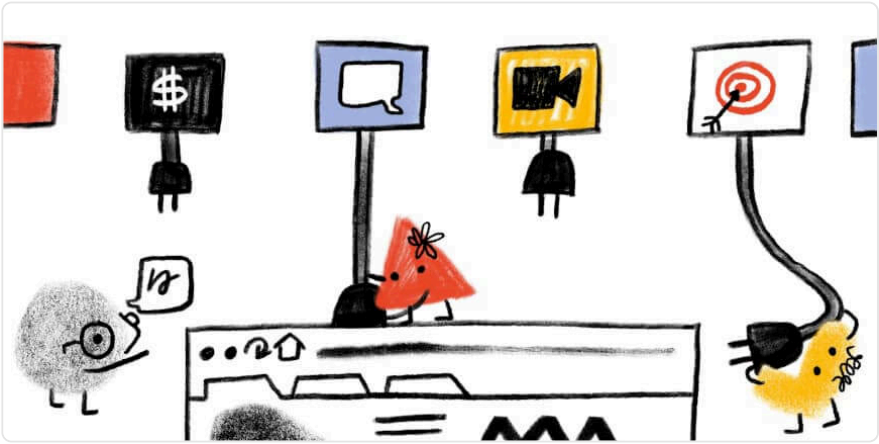
著者

Nimesh Vadgama - VN

nimeshgit ops-ml-architect https://ops-ml-architect.blogspot.com/

NimeshはAI・MLアナリティクス、オペレーション、ビジネスプロセスに焦点を当て、デジタルトランスフォーメーションと自動化ソリューションを提供しています。

部 I 章 3
サードパーティ



Muhammad Jazlan と Muhammad Abu Bakar Aziz によって書かれた。
Barry Pollard によってレビュ。
Muhammad Jazlan による分析。
Barry Pollard 編集。
Sakae Kotaro によって翻訳された。

はじめに

サードパーティはウェブ上にあまねく存在しています。ウェブサイト開発者は、広告、アナリティクス、ソーシャルメディア連携、決済処理、コンテンツ配信などの主要機能を実装するためにサードパーティに依存しています。このモジュール型のアプローチは、豊富な機能の効率的かつ迅速なデプロイを可能にします。しかし、プライバシー、セキュリティ、パフォーマンスに関する潜在的な懸念をもたらします。今年新たに、使用されているコンセントフレームワークやシグナルを受け取るサードパーティを含め、ユーザーのコンセント選択がウェブ上のサードパーティ間でどのように伝達されるかを分析しています。

本章では、ウェブにおけるサードパーティの使用パターンについて実証的な分析を行います。以下の観点から調査します：

- **普及率：** どれだけのウェブサイトがサードパーティを使用しているか、またその割合

- **リソースタイプ：** サードパーティが取る形式（画像、JavaScript、フォントなど）
- **機能カテゴリ：** 広告ネットワーク、アナリティクス、CDN、動画プロバイダー、タグマネージャーなど
- **統合方法：** サードパーティがページに直接または間接的にどのようにロードされるか
- **コンセントインフラ：** どのサードパーティがコンセントシグナルを転送し、それらの転送が実際にどのように行われているか

定義

まず、分析全体で使用される定義と用語を確認します。

サイトとページ

本章では、これまでの年と同様に、「サイト」という用語を特定のドメインの登録可能な部分を表すために使用します。これはしばしば*拡張トップレベルドメインプラス1*（eTLD+1）と呼ばれます。例えば、URL `https://www.bar.com/` のeTLD+1は `bar.com` であり、URL `https://foo.co.uk` のeTLD+1は `foo.co.uk` です。「ページ」（またはウェブページ）とは、固有のURL、より具体的には特定のURLに存在するドキュメント（例えばHTMLやJavaScript）を意味します。

サードパーティとは？

以前のバージョンとの比較を可能にするため、Web Almanacの前版で使用されたサードパーティの定義に従います。

*サードパーティ*とは、サイトオーナー（ファーストパーティとも呼ばれる）とは異なるエンティティです。サイトオーナーが直接実装・提供していないサイトの側面を含みます。より正確には、サードパーティのコンテンツはユーザーが最初に訪問したサイトではなく、別のサイトからロードされます。ユーザーが `example.com`（ファーストパーティ）を訪問し、`example.com` が `awesome-cats.edu` から面白い猫の画像を含んでいる（例えば `<img>` タグを使って）とします。このシナリオでは、`awesome-cats.edu` はユーザーが最初に訪問したものではないため、サードパーティです。ただし、ユーザーが直接 `awesome-cats.edu` を訪問した場合、`awesome-cats.edu` はファーストパーティとなります。

分析では、HTTP Archiveデータセットの少なくとも5つの固有ページでリソースが見つかる

ドメインから発生するサードパーティのみを含めました。

サードパーティのコンテンツがファーストパーティドメインから直接提供される場合、ファーストパーティのコンテンツとしてカウントされます。例えば、セルフホストされたアナリティクススクリプト、CSS、またはフォントはファーストパーティのコンテンツとしてカウントされます。同様に、サードパーティドメインから提供されるファーストパーティのコンテンツはサードパーティのコンテンツとしてカウントされます。一部のサードパーティは異なるサブドメインからコンテンツを提供しています。ただし、サブドメインの数に関わらず、単一のサードパーティとしてカウントされます。

さらに、サードパーティがファーストパーティに偽装されることがますます一般的になっています。これを可能にする2つの主要な技術があります：

- **CNAMEクローキング**は、CNAMEレコードを使用してサードパーティのコンテンツがファーストパーティドメインから来ているように見せる技術です。本分析では、CNAMEクローキングされたサービスはファーストパーティとして扱います。

- **サーバーサイドトラッキング**は、サイトオーナーがトラッカーをファーストパーティとして組み込み、すべてのリクエストをファーストパーティドメイン経由でルーティングし、トラッカーをファーストパーティのように見せる新興のトレンドです。例えば、ウェブサイト `www.example.com` がサーバーサイドGoogle Tag ManagerをGoogle Analyticsとともに組み込み、サブドメイン `sst.example.com` をクロークしてGoogle Tag Managerコンテナにリクエストを送信する場合があります。このようにして、サードパーティへのリクエストはユーザーのブラウザではなく、タグマネージャーのサーバーから発生します。

分析では、サードパーティの通信はサーバー間で行われ、クライアントサイドのHTTP Archiveデータでは直接観測できないため、このようなケースをファーストパーティのインタラクションとして扱います。その結果、私たちの測定値はウェブ上のサードパーティの実際の普及率の下限を表しています。

カテゴリ

前述の通り、サードパーティは動画の埋め込み、広告の配信、ソーシャルメディアサイトのコンテンツの埋め込みなど、様々なユースケースで使用できます。昨年と同様に、データセ

ットで観察されたサードパーティをカテゴリ分けするために、Patrick Hulce[10]によるThird-Party Web[11]リポジトリを使用します。このリポジトリは以下のカテゴリでサードパーティを分類しています：

- **Ad（広告）：** これらのスクリプトは広告ネットワークの一部で、広告の配信または計測を行います。
- **Analytics（アナリティクス）：** これらのスクリプトはユーザーとその行動を計測またはトラッキングします。追跡対象によって影響範囲は大きく異なります。
- **CDN：** これらは公開CDNや私有CDNを通じて配信される、公開ホストされたオープンソースライブラリ（例：jQuery）と私的なCDN使用の混合です。
- **Content（コンテンツ）：** これらのスクリプトはコンテンツプロバイダーまたは出版特有のアフィリエイトトラッキングからのものです。
- **Customer Success（カスタマーサクセス）：** これらのスクリプトはチャットや問い合わせソリューションを提供するカスタマーサポート/マーケティングプロバイダーからのものです。これらのスクリプトは一般的に重量が大きいです。
- **Hosting（ホスティング）：** これらのスクリプトはウェブホスティングプラットフォーム（WordPress、Wix、Squarespaceなど）からのものです。
- **Marketing（マーケティング）：** これらのスクリプトはポップアップ/ニュースレターなどを追加するマーケティングツールからのものです。
- **Social（ソーシャル）：** これらのスクリプトはソーシャル機能を有効にします。
- **Tag Manager（タグマネージャー）：** これらのスクリプトは多くの他のスクリプトをロードし、多くのタスクを開始する傾向があります。
- **Utility（ユーティリティ）：** これらのスクリプトは開発者向けのユーティリティ（APIクライアント、サイト監視、不正検出など）です。
- **Video（動画）：** これらのスクリプトは動画プレーヤーとストリーミング機能を有効にします。
- **Consent provider（コンセントプロバイダー）：** これらのスクリプトはサイトがユーザーのコンセントを管理することを可能にします（例：一般データ保護規則[12]のコンプライアンスのため）。クッキーコンセント ポップアップとも呼ばれ、通常はクリティカルパスでロードされます。
- **Other（その他）：** これらは正確なカテゴリや帰属のない共有オリジン経由で配

10. *https://x.com/patrickhulce*
11. *https://github.com/patrickhulce/third-party-web/#third-parties-by-category*
12. *https://wikipedia.org/wiki/General_Data_Protection_Regulation*

信される雑多なスクリプトです。

`Content-Type`

`Content-Type` HTTPヘッダーを使用して、サードパーティリソースをスクリプト、HTMLコンテンツ、JSONデータ、プレーンテキスト、画像などの異なるタイプに分類します。これにより、ウェブサイト全体で配信されるサードパーティリソースの構成を分析できます。

普及率

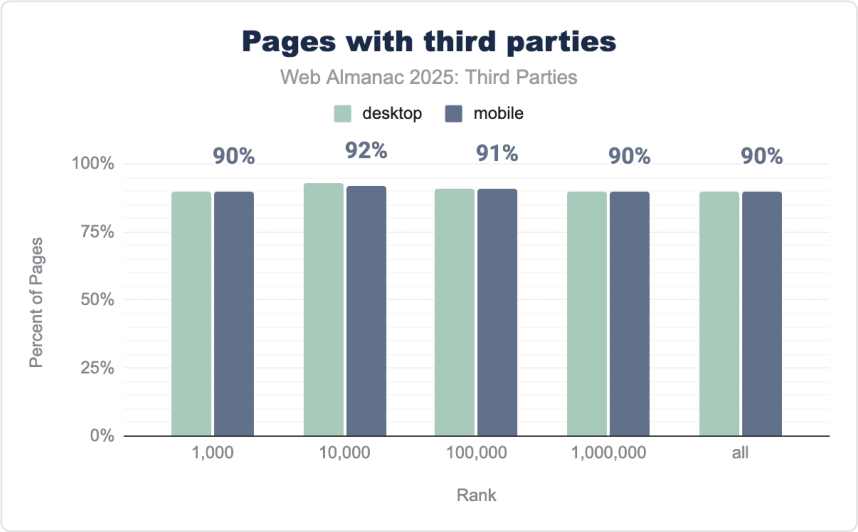


図3.1. 1つ以上のサードパーティを使用しているページの割合。

前年[13]と比較して、ウェブサイト全体で1つ以上のサードパーティを使用しているページの割合にわずかな減少が見られます。ただし、この減少にもかかわらず、1つ以上のサードパーティを持つページの割合は90%以上を維持しています。

13. *https://almanac.httparchive.org/ja/2024/third-parties#普及率*

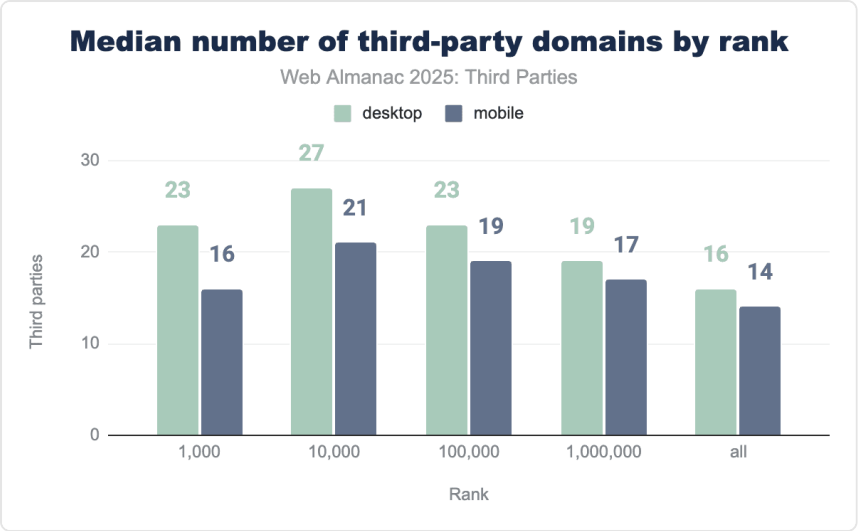


図3.2. ランク別サードパーティ数の分布。

前年と比較して、全ウェブサイトランクにわたってサードパーティドメインの中央値が大幅に減少しており、特に低ランクのウェブサイトで大きな減少が見られます。

この減少にはいくつかの要因が考えられます。第一に、サードパーティはCNAMEクローキングやサーバーサイドトラッキングを通じてますます隠蔽されており、クライアントサイドの計測での可視性が低下する可能性があります。第二に、HTTP Archiveのクローラーはウェブページとのインタラクションやページのスクロールを行わないため、遅延ロードにより一部のサードパーティが正しくロードされない場合があります。その結果、観察されるサードパーティリクエストが少なくなる可能性があります。

また、デスクトップページはモバイルページよりも一般的に多くのサードパーティを含んでいることも確認されています。

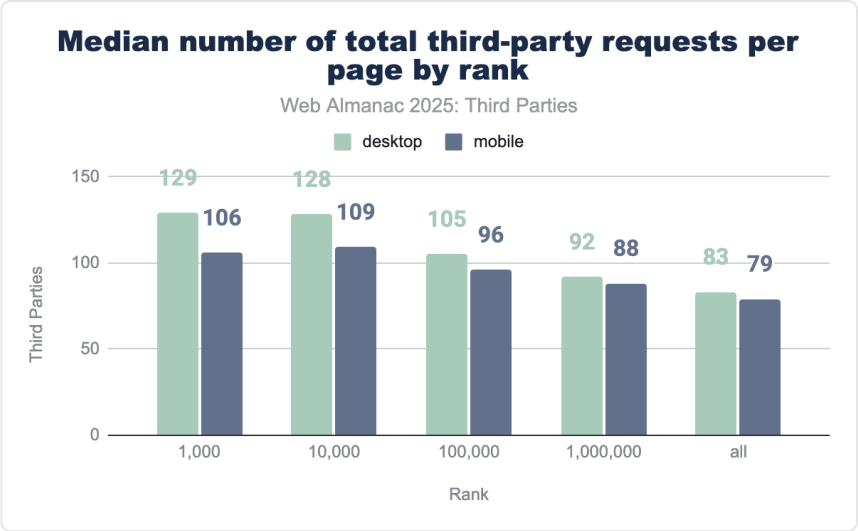


図3.3. ランク別ページあたりのサードパーティリクエスト数の分布。

低ランクのウェブサイトはより多くのサードパーティリクエストをロードします。トップ1,000はデスクトップで129リクエスト、モバイルで106リクエストの中央値を持ち、全サイトのデスクトップ83リクエスト、モバイル79リクエストと比較しています。

前年比で、すべてのランクにわたってサードパーティリクエストが増加しています。トップ1,000サイトは2024年と比較して[14]デスクトップで15リクエスト、モバイルで15リクエストの増加を示しており、より広いデータセットではデスクトップで5リクエスト、モバイルで5リクエスト増加しました。この上昇トレンドは、先ほど観察したユニークなサードパーティドメイン数の減少にもかかわらず起きており、個々のサードパーティがページあたりより多くのリクエストを送信していることを示しています。

14. *https://almanac.httparchive.org/ja/2024/third-parties#fig-3*

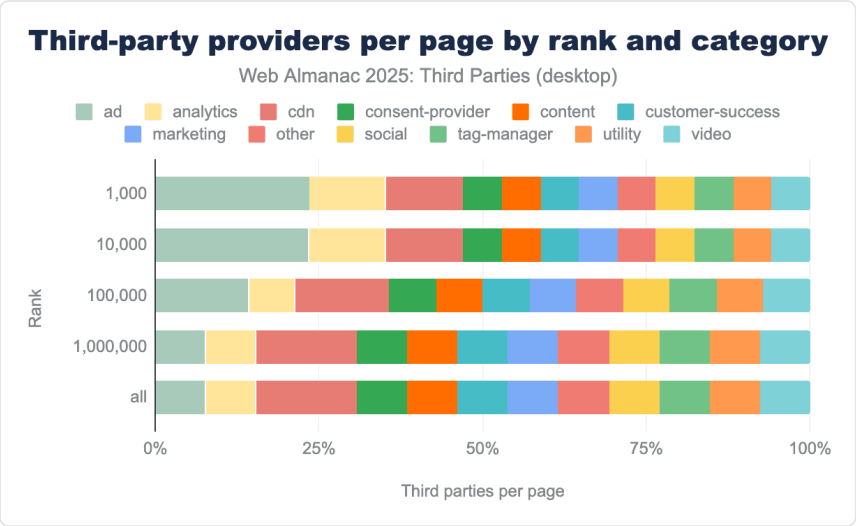


図3.4. ランク別サードパーティリクエストカテゴリの分布。

棒グラフはランクとカテゴリ別のページあたりのサードパーティプロバイダーの中央値を示しています。前版ではこの分析はランクとカテゴリ別のページあたりのサードパーティドメイン数に焦点を当てていましたが、今年はユニークなサードパーティプロバイダーの数を計測しており、全体的に低い数値となっています。今年のトップカテゴリは `ad`、`analytics`、`cdn` です。

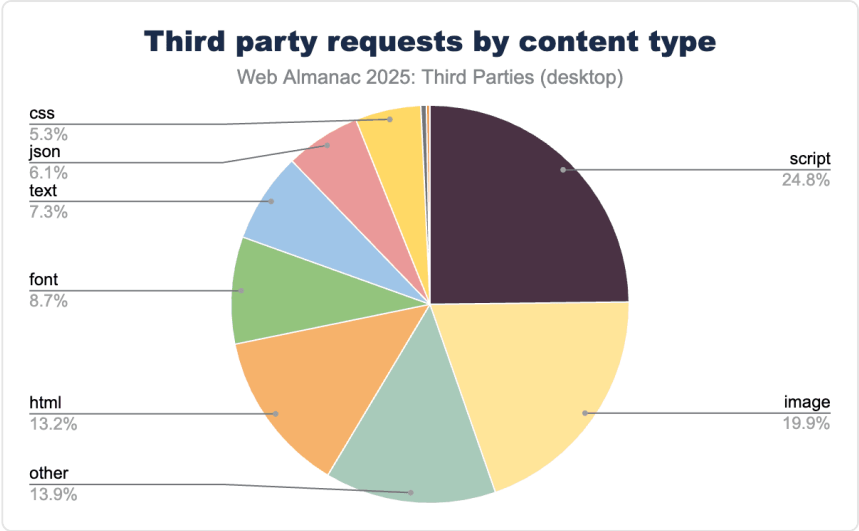


図3.5. ランク別サードパーティリクエストタイプの分布。

チャートはサードパーティリクエストが `script`、`image`、`other` カテゴリによって支配されていることを示しています。`script`、`image`、`other` の合計は全サードパーティリクエストのコンテンツタイプの半数以上を占めています。このパターンは `script`、`image`、`other` もトップリクエストタイプとして特定した2024年版[15]と一致しており、昨年からほとんど変化がないことを示しています。

15. *https://almanac.httparchive.org/ja/2024/third-parties#fig-5*

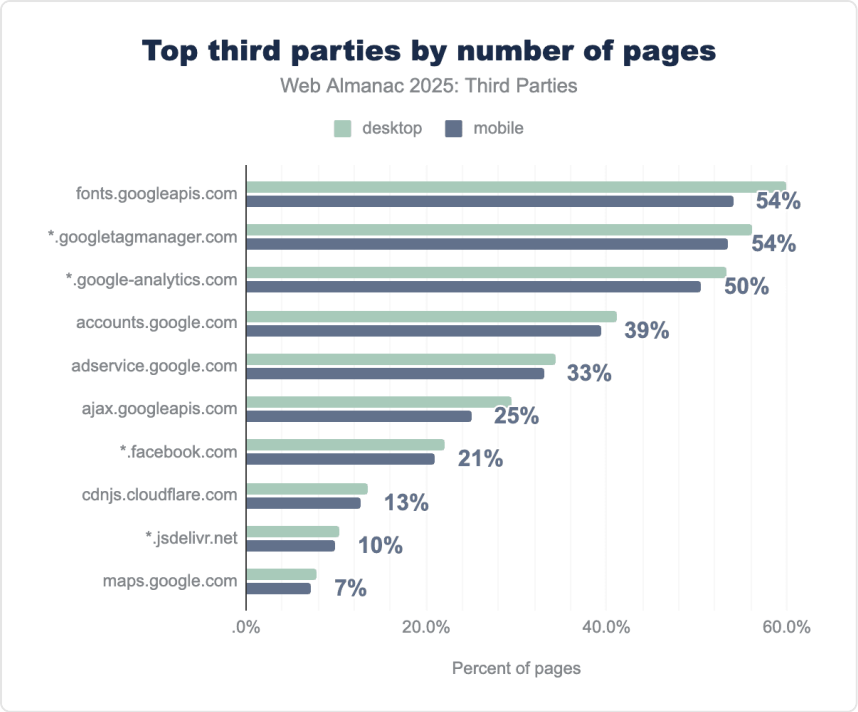


図3.6. ページ数別トップサードパーティ。

トップ10のサードパーティドメインは `fonts.googleapis.com`、`googletagmanager.com`、`google-analytics.com`、`accounts.google.com`、`adservice.google.com` を含むGoogle所有のサービスが独占しています。Metaの `facebook.com` はトップ10で唯一のGoogle以外のドメインで、21%のページで7位にランクインしています。

サードパーティ間のコンセント伝達

このセクションでは、異なるサードパーティがウェブ全体でユーザーコンセントをどのように転送しているかを調査します。先行研究[16]では、サードパーティがコンセント情報の伝達に業界標準フレームワークに依存することが多いことが示されています。分析では、IABの3つのコンセント標準、すなわち透明性とコンセントフレームワーク（TCF）[17]、CCPAフレームワーク[18]、およびグローバルプライバシープロトコル（GPP）[19]に主に焦点を当てていま

16. *https://petsymposium.org/popets/2024/popets-2024-0120.pdf*
17. *https://iabeurope.eu/transparency-consent-framework/*
18. *https://iabtechlab.com/standards/ccpa/*
19. *https://iabtechlab.com/gpp/*

す。

これらのフレームワークはコンセント情報がウェブサイトとサードパーティ間でどのようにエンコードされ共有されるかを定義しています。データセットで観察されたサードパーティの中でどのコンセント標準が最も普及しているかを特定することから始めます。サードパーティが使用するフレームワークを判定するために、リクエストURLの特定パラメータの存在に依存しています。異なる標準の詳細は以下の通りです：

- **TCF標準**: IAB TCFで指定されているように、サードパーティリクエストに `gdpr` または `gdpr_consent` パラメータが含まれているかを確認することでTCFフレームワークの使用を特定します。

- **GPP標準**: `gpp` と `gpp_sid` パラメータの存在を確認することでGPPフレームワークの使用を特定します。

- **USP標準と非USP標準**: IAB CCPAフレームワークで定義されているように、リクエストが `us_privacy` パラメータを送信しているかを確認することでUSP標準の使用を特定します。また、先行研究[20]で特定された非標準パラメータを通じて送信されるコンセント文字列を検出することで、非標準USP標準の使用も特定します。

ウェブサイトランク、サードパーティカテゴリ、およびコンセントを受け取る最も頻繁に観察されるサードパーティにわたってコンセントシグナルの普及率を分析します。

20. *https://petsymposium.org/popets/2024/popets-2024-0120.pdf*

異なるランクにわたるコンセントシグナルの普及率

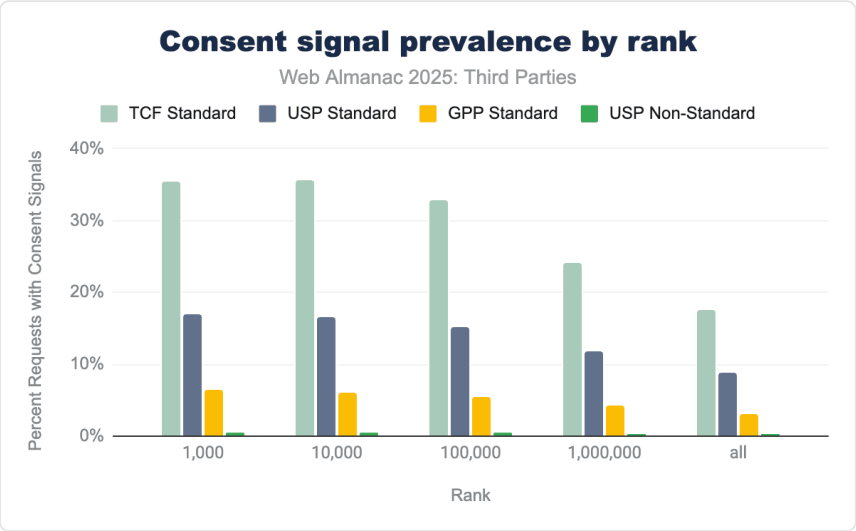


図3.7. ランク別コンセントシグナルの普及率。

TCF標準が主要なコンセント標準であり、特に低ランクサイトでは全サイトの18%に対して36%に達しています。この高い採用率はGDPR下での強力なオプトインコンセント要件と一致しています。USP標準は2番目に普及しており、ランク全体で9〜17%の採用率です。これはCCPAに対応して導入されたIAB CCPAコンセントフレームワークの使用を反映しています。GPPの採用は管轄区域をまたいでコンセントフレームワークを統一するという目標にもかかわらず、3〜6%と最小限に留まっています。

異なるカテゴリにわたるコンセント標準の分布

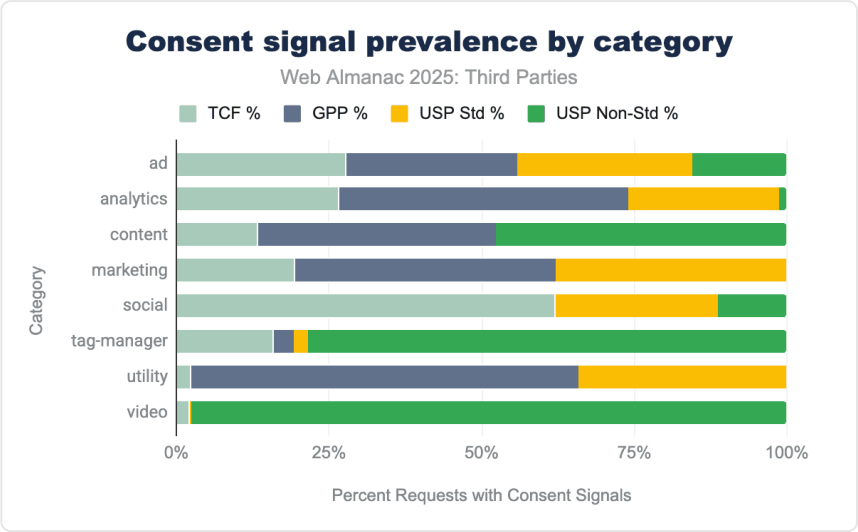


図3.8. カテゴリ別コンセントシグナルの普及率。

異なるサードパーティカテゴリにわたって異なるコンセント標準の選好が見られます。例えば、Socialサービスは最高のTCF採用率を示している一方、広告ベンダーはGPP、USP標準、そして小さなTCFシェアのより均衡した組み合わせを採用しています。さらに、Analyticsベンダーは主にGPPを採用しています。

コンセントを受け取るトップサードパーティ

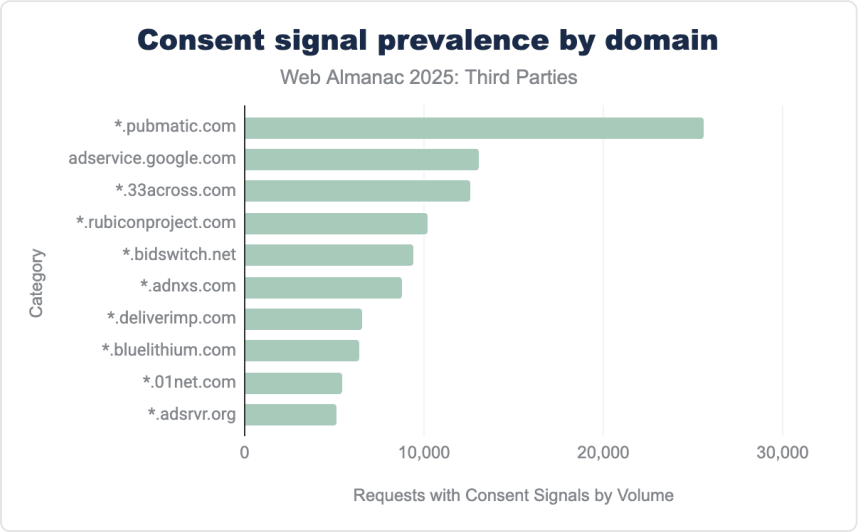


図3.9. ドメイン別コンセントシグナルの普及率。

トップランクのウェブサイトの中で、`pubmatic.com` が最も多くのコンセントシグナルを受け取り、`adservice.google.com` が2位です。最も多くのコンセントシグナルを受け取るドメインの大多数は広告とアドテクベンダー（広告取引所、DSP、広告サーバー）です。多くの管轄区域でサードパーティの広告やアナリティクスプロバイダーが広告やその他の目的でユーザーデータを使用する前にユーザーのコンセントを取得しなければならないことを考えると、これは直感的に理解できます。

インクルージョン

先ほどの例を振り返ると、`example.com`（ファーストパーティ）が `awesome-cats.edu`（`<img>` タグによるサードパーティ）から画像を含めることができます。この画像のインクルージョンは直接インクルージョンとみなされます。しかし、もし画像が `XMLHttpRequest` を介してサイト上のサードパーティスクリプトによってロードされた場合、その画像のインクルージョンは間接インクルージョンとみなされます。間接的にインクルードされたサードパーティはさらに追加のサードパーティを含める場合があります。例えば、サイトに直接インクルードされたサードパーティスクリプトが、さらに別のサードパーティスクリプトを含める場合があります。本章では、サードパーティのインクルージョンチェーンの深さについて基本的な分析を行います。

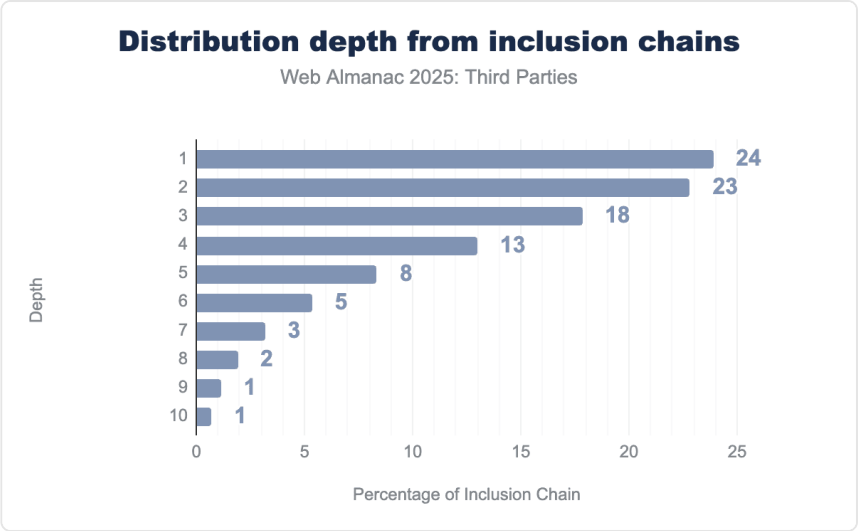


図3.10. サードパーティインクルージョンチェーンの中央深度。

インクルージョンチェーンの中央深度は3であり、サードパーティの大多数がウェブページ上で少なくとも別のサードパーティを含めていることを意味します。インクルージョンチェーンの最大深度は2,285です。

結論

私たちの調査結果は、ウェブ上のサードパーティの遍在性と集中化が進んでいることを示しています。10のウェブページのうち9つ以上が1つ以上のサードパーティを含んでいます。前年と比較してユニークなサードパーティドメインの中央値は減少していますが、サードパーティからのリクエストの総数は大幅に増加しており、個々のベンダーがページあたりより多くのリクエストを送信していることを示しています。

コンセント標準に関しては、TCFが全ウェブサイトランクにわたって主要なコンセント標準です。個々のサードパーティの中では、`pubmatic.com`、`adservice.google.com` およびその他のアドテクドメインが最も多くのコンセントシグナルを受け取っています。

最後に、CNAMEクローキングやサーバーサイドトラッキングなどの難読化技術の使用の増加は、クライアントサイドの計測でのサードパーティの可視性を低下させており、私たちの調査結果が実際の普及率の下限を表していることを示しています。

著者

Muhammad Jazlan

jazlan01

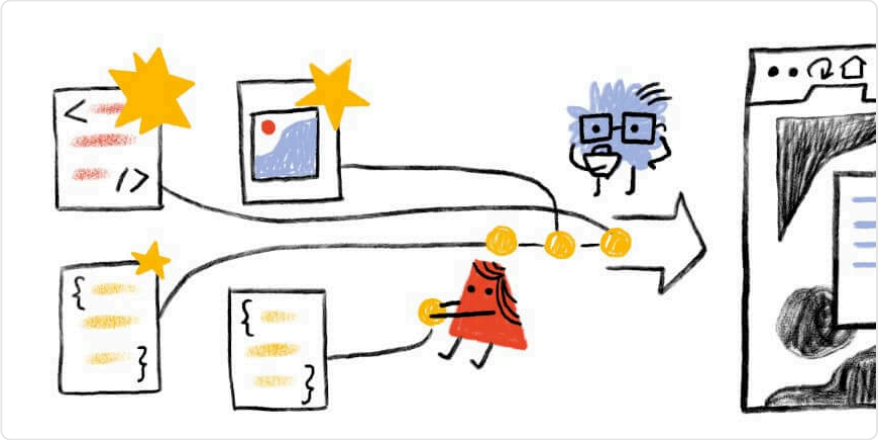
Muhammad Jazlanはカリフォルニア大学デービス校のコンピュータサイエンス博士課程2年生です。ウェブにおけるトラッキングの計測、検出、および軽減を研究テーマとしています。

Muhammad Abu Bakar Aziz

abubakaraziz aziz313f

Muhammad Abu Bakar Azizはボストンにあるノースイースタン大学のコンピュータサイエンス博士候補者です。ウェブプライバシーを研究テーマとしており、特にサードパーティやオンライン広告主がCCPAやGDPRなどのプライバシー法をどのように遵守しているかを実証的に計測しています。

部 I 章 4
生成AI



***Christian Liebel*、 *Yash Vekaria* と *Jonathan Pagel* によって書かれた。**
Thomas Steiner、 *Umar Iqbal* と *Maxim Salnikov* によってレビュ。
Jonathan Pagel、 *Christian Liebel* と *Thomas Steiner* による分析。
Barry Pollard 編集。
Sakae Kotaro によって翻訳された。

はじめに

2022年11月30日、OpenAIはChatGPTを公開しました[21]。このサービスは *生成的人工知能*（生成AI）を研究室から数百万人の日常生活へ[22]と一気に引き上げました。このリリースはアプリケーションとウェブがどうあるべきかというユーザーの期待を根本から変えました。さらに、付随する *アプリケーションプログラミングインターフェース*（API）はソフトウェア開発者に、アプリケーションを劇的にスマートにする強力なツールをもたらしました。

21. *https://openai.com/index/chatgpt/*
22. *https://openai.com/index/how-people-are-using-chatgpt/#:~:text=Given%20the%20sample%20size%20and%20700%20million%20weekly%20active%20users%20of%20ChatGPT*

図4.1. ChatGPTの週間アクティブユーザー数。

生成AIは、テキスト、ソースコード、画像、動画、音声、音楽など人間が理解できるコンテンツの処理と生成に特化した専門分野です。 *大規模言語モデル*（LLM）はこの分野の重要なコンポーネントです。膨大なテキストデータで学習されたLLMは自然言語を解釈・生成し、開発者が初めて人間言語を効果的に処理できるようにすることでソフトウェアアーキテクチャを拡張しています。最近では生成AI機能がWindows、Office、Photoshopなどの確立されたアプリケーションにも統合されています。

生成AIがますます普及する中、この章ではウェブにおける生成AIの新興トレンドを検討します。具体的には、「組み込みAI」と「Bring Your Own AI」アプローチによって実現するローカル生成AIの使用、`llms.txt`を介した生成AIコンテンツの発見可能性、そしてコンテンツ作成とソースコードへの生成AIの影響（*AIフィンガープリント*）に焦点を当てます。

データソース

この章では、HTTP Archiveのデータセットだけでなく、npm[23]のダウンロード統計、およびChrome Platform Status[24]や他の研究者が提供するデータも使用しています。

特に断りのない限り、方法論で説明されている2025年7月のHTTP Archiveクロールを参照しています。APIの採用を検出するために、コード内の特定のAPI呼び出しの存在についてウェブサイトをスキャンしました。これはその存在を示すものであり、実行時の実際の使用状況を必ずしも示すものではありません。Chrome Platform Statusの使用データは常に、すべてのリリースチャンネルとプラットフォームにわたって、特定のAPIを少なくとも1回使用するGoogle Chromeのページロードの割合を指しています。

技術概要

このセクションでは、クラウドベースのAIモデルとローカルAIモデルの違いを説明し、これらのアプローチの長所と短所について議論した後、ローカル技術を詳しく検討します。

23. *https://www.npmjs.com/*
24. *https://chromestatus.com/*

クラウド対ローカル

ほとんどの人は、OpenAI[25]、Microsoft Foundry[26]、AWS Bedrock[27]、Google Cloud AI[28]、またはDeepSeekプラットフォーム[29]などのクラウドサービスを通じて生成AIを使用しています。これらのプロバイダーは膨大な計算リソースとストレージ容量にアクセスできるため、いくつかの重要な利点を提供しています：

- **高品質なレスポンス**: モデルは非常に有能で強力です。
- **高速な推論時間**: 強力なサーバーでレスポンスが素早く生成されます。
- **最小限のデータ転送**: AIモデル全体ではなく、入出力データのみを転送する必要があります。
- **ハードウェア非依存**: クライアントのハードウェアや計算リソースに関わらず動作します。

しかし、クラウドベースのモデルには欠点もあります：

- **接続性**: 安定したインターネット接続が必要です。
- **信頼性**: ネットワーク遅延、サーバーの可用性、容量制限の影響を受けます。
- **プライバシー**: データをクラウドサービスに転送する必要があり、潜在的なプライバシーの懸念が生じます。ユーザーデータが将来のモデルの反復学習に使用されているかどうかは、しばしば不明確です。
- **コスト**: 通常、サブスクリプションやAPI使用料が必要であり、ウェブサイト作者は推論のコストを負担しなければならないことが多いです。

ローカルAI技術

クラウドベースシステムの限界は、Web AIと呼ばれる[30]ローカルAI技術を介して推論をクライアントに移行することで対処できます。モデルはユーザーのシステムにダウンロードされるため、その *重み*（内部モデルパラメーター）を秘密にすることはできません。そのため、このアプローチは主にオープンウェイトモデルと組み合わせて使用されますが、オープンウェイトモデルは商用のクラウドベースのクローズドウェイトモデルと比べて一般的に性

25. *https://openai.com/api/*
26. *https://ai.azure.com/*
27. *https://aws.amazon.com/de/bedrock/*
28. *https://cloud.google.com/products/ai*
29. *https://platform.deepseek.com/*
30. *https://developer.chrome.com/blog/io24-web-ai-wrapup#:~:text=Web%20AI%20is%20a%20set%20of%20technologies%20and%20techniques%20to%20use%20machine%20learning%20(ML)%20models%2C%20client%2Dside%2C*

能が低い[31]です。Epoch AIによると[32]、オープンウェイトモデルは平均して約3か月、最先端の性能に遅れをとっています。

図4.2. オープンウェイトモデルが最先端性能に遅れをとる平均期間。

World Wide Web Consortium[33]（W3C）のWeb Machine Learning[34]コミュニティグループとワーキンググループは、AIを「ファーストクラスのウェブ市民」にするためにこの移行の標準化を積極的に進めています。この取り組みは2つの主要なアーキテクチャ方向に従っています：*Bring Your Own AI* と *組み込みAI* です。

Bring Your Own AI

Bring Your Own AI（BYOAI）アプローチでは、開発者がモデルをユーザーに配信する責任を担います。ウェブアプリケーションは特定のモデルバイナリをダウンロードし、ローカルハードウェア上の低レベルAPIを使用して実行します。これにより、ユースケースに特化した高度に専門化されたモデルを実行できますが、大きな帯域幅も必要とします。

AIの推論をローカルで実行するために使用できる3つの処理ユニット[35]があります：

- *中央処理装置*（CPU）：応答が速く、低遅延のAIワークロードに最適。
- *グラフィックス処理装置*（GPU）：高スループットで、AI加速デジタルコンテンツ作成に最適。
- *ニューラル処理装置*（NPU）または *テンソル処理装置*（TPU）：低消費電力で、継続的なAIワークロードとバッテリー寿命のためのAIオフロードに最適。

ローカルAI推論を促進する3つの主要なAPIはWebAssembly（CPU）、WebGPU（GPU）、WebNN（CPU、GPU、NPU）です。

WebAssemblyとWebGPUの使用がAI活動を確認するわけではないことに注意することが重要です。これらは複雑な計算、3Dビジュアライゼーション、またはゲームなどのタスクに頻繁に使用される汎用APIです。

31. *https://www.vellum.ai/llm-leaderboard*
32. *https://epoch.ai/data-insights/open-weights-vs-closed-weights-models*
33. *https://www.w3.org/*
34. *https://webmachinelearning.github.io/*
35. *https://www.w3.org/2024/01/webevolve-series-events/media/slides/hu-ningxin.pdf#page=4*

WebAssembly

WebAssembly[36]はウェブのバイトコードとして機能します。C++やRustなどのさまざまなプログラミング言語で書かれたコードをWebAssemblyにコンパイルできます。開発者がユーザーのCPUでブラウザのスクリプティングエンジンによって実行される最適化された高性能コードを書くことができます。

WebAssemblyは2017年以来すべての主要なブラウザエンジンに実装されており、広範なブラウザサポートを持っています。

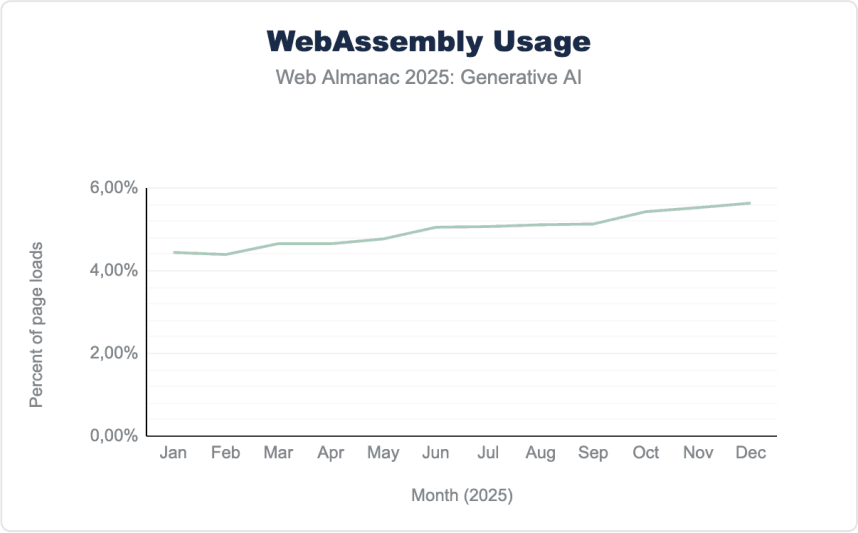


図4.3. Chrome Platform Statusデータによる2025年のWebAssembly使用状況。

Chrome Platform Statusデータ[37]によると、WebAssemblyの使用率は2025年1月の4.44%のページロードから2025年12月の5.64%へと27%の線形成長を経験しました。2024年1月、WebAssemblyはページロードの3.37%でのみ有効でした。

WebGPU

WebGPU[38]はWebGL[39]の近代的な後継であり、現代のGPUの機能をウェブに公開するために設計されています。厳密にグラフィックス用であったWebGLとは異なり、WebGPUは *コンピュートシェーダー* のサポートを提供し、グラフィックスカード上での汎用コンピューティングを可能にします。これにより、開発者はAIモデルが必要とする大規模な並列計算をユー

36. *https://developer.mozilla.org/docs/WebAssembly*
37. *https://chromestatus.com/metrics/feature/timeline/popularity/2237*
38. *https://developer.mozilla.org/docs/Web/API/WebGPU_API*
39. *https://developer.mozilla.org/docs/Web/API/WebGL_API*

ザーのグラフィックスカード上で直接実行できます。

WebGPUはブラウザでのAIワークロード実行の標準的な基盤となっています。2025年11月のFirefox 141のリリースにより、WebGPUはすべての主要なブラウザエンジンで利用可能になりました[40]（Chromium、Gecko、WebKit）。

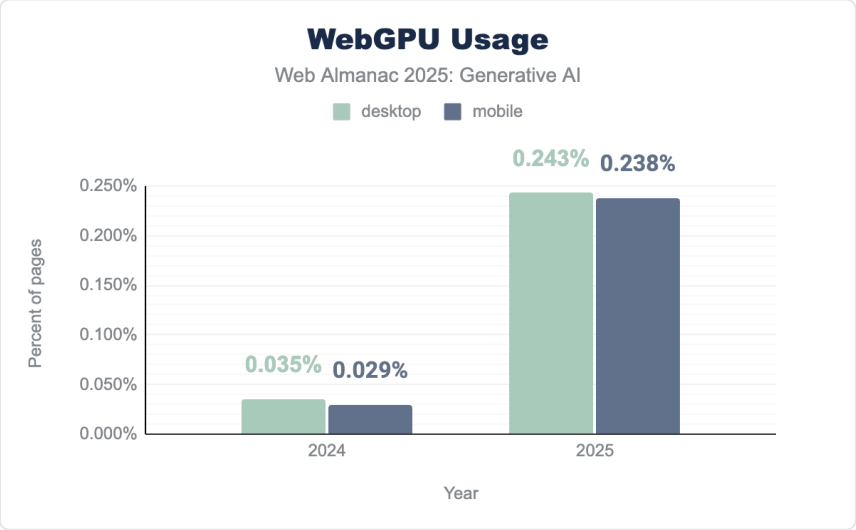


図4.4. 2025年のWebGPU使用状況。

2025年7月のHTTP ArchiveデータクロールはAPIがすべてのデスクトップサイトの0.243%とモバイルサイトの0.238%で使用されていることを示しています。全体的にはまだ小さいながらも、2024年7月クロールと比較してデスクトップで591%（0.035%から増加）、モバイルで709%（0.029%から増加）という大幅な増加を示しています。Chrome Platform Statusデータ[41]は、2025年を通じて147%増加したページロードあたりのアクティベーションの指数的成長を示唆しています。

WebNN

Web Neural Network API[42]（WebNN）は機械学習に特化した専用APIです。WebMLワーキンググループによって仕様策定され、このAPIはW3C勧告トラック—ウェブ標準となるための正式なプロセス—に位置づけられています。

WebNNはハードウェア非依存の抽象化レイヤーとして機能し、ブラウザがデバイス上で利

40. *https://web.dev/blog/webgpu-supported-major-browsers*
41. *https://chromestatus.com/metrics/feature/timeline/popularity/3888*
42. *https://webnn.io/en*

用可能な最も効率的なハードウェアに処理をルーティングできるようにします。WebAssembly（CPUのみ）とWebGPU（GPUのみ）とは対照的に、WebNNはCPU、GPU、NPUで計算を実行できます。ネイティブに近い実行速度を達成できます。

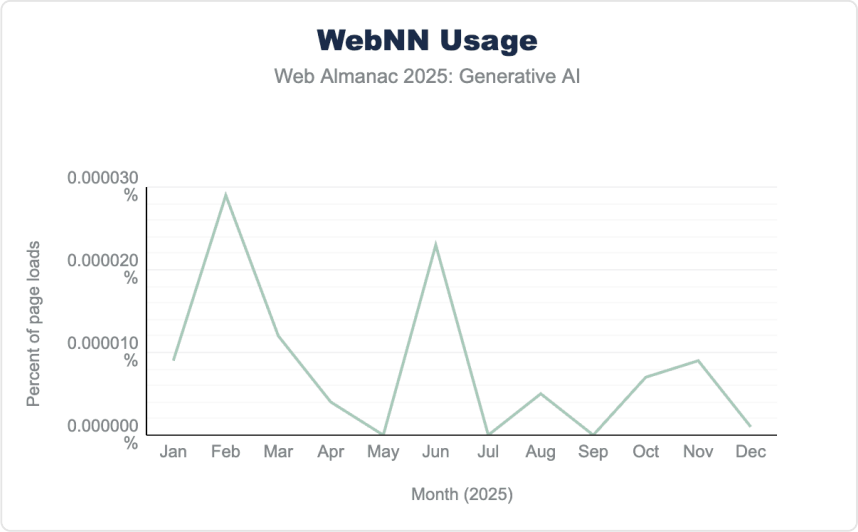


図4.5. Chrome Platform Statusデータによる2025年のWebNN使用状況。

WebNNは現在活発に開発中であり、ChromeOS、Linux、macOS、Windows、AndroidのChromiumベースのブラウザ[43]でフラグの後ろに実装されています。2025年11月、FirefoxがAPIを正式にサポートする2番目のエンジンとして参加しました[44]。WebNNがまだ実験的なAPIであることを考えると、Chrome Platform Statusデータ[45]によれば使用数は現在非常に低く、変動が大きく、2025年2月の最大アクティベーション率は0.000029%です。

ランタイム：ONNX RuntimeとTensorflow.js

ONNX Runtime[46]（Microsoftが開発）とTensorflow.js[47]（Googleが開発）は、ブラウザ上でAIモデルを直接実行するための主要なランタイムの2つです。これらのランタイムはWebAssembly、WebGPU、WebNNなどの低レベル技術の複雑さを抽象化します。

TensorFlow.jsはTensorFlowエコシステムと緊密に統合されており、学習と推論の両方をサポートしている一方、ONNX RuntimeはONNX標準を使用したクロスプラットフォーム推論に焦点を当て、複数のフレームワークからのモデルをクライアントサイドで実行できるように

43. *https://webnn.io/en/api-reference/browser-compatibility/api*
44. *https://github.com/mozilla/standards-positions/issues/1215#issuecomment-3520278819*
45. *https://chromestatus.com/metrics/feature/timeline/popularity/5023*
46. *https://onnxruntime.ai/docs/tutorials/web/*
47. *https://www.tensorflow.org/js*

しています。

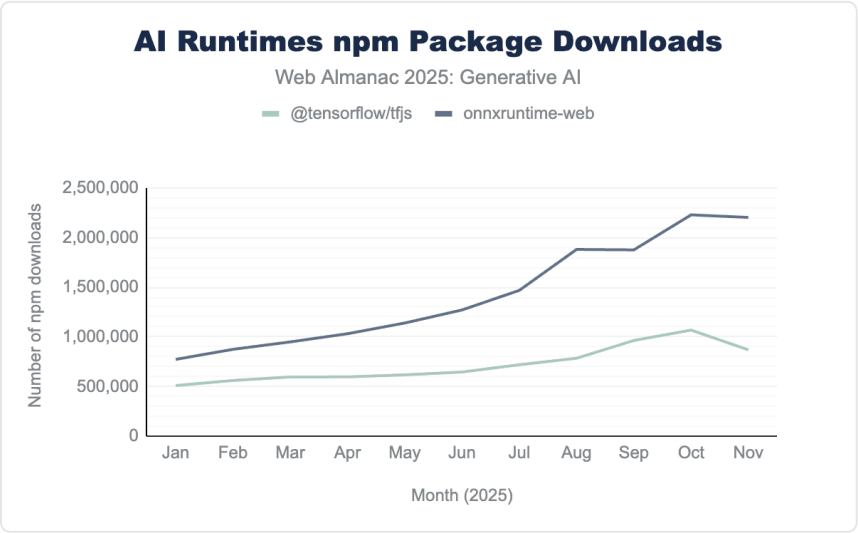


図4.6. `@tensorflow/tfjs` *と* `onnxruntime-web` *のnpmパッケージダウンロード数。*

2025年1月から11月にかけて、ONNX Runtimeのnpmパッケージダウンロード数は[48]185%増加し、TensorFlow.jsのダウンロード数[49]は70%増加し、ブラウザベースのAIへの開発者の強い関心と高まりを示しています。

ライブラリ：WebLLMとTransformers.js

MLC AI[50]研究チームが開発したWebLLM[51]は、LLMに特化した高性能なブラウザ内推論エンジンです。Llama[52]、Phi[53]、Gemma[54]、Mistral[55]などの様々なオープンウェイトLLMをブラウザ上で直接実行できます。現在、WebLLMは推論にWebGPUを使用しています。

48. *https://npm-stat.com/charts.html?package=onnxruntime-web&from=2025-01-01&to=2025-11-30*
49. *https://npm-stat.com/charts.html?package=%40tensorflow%2Ftfjs&from=2025-01-01&to=2025-11-30*
50. *https://mlc.ai/*
51. *https://webllm.mlc.ai/*
52. *https://www.llama.com/*
53. *https://azure.microsoft.com/en-us/products/phi*
54. *https://deepmind.google/models/gemma/*
55. *https://mistral.ai/*

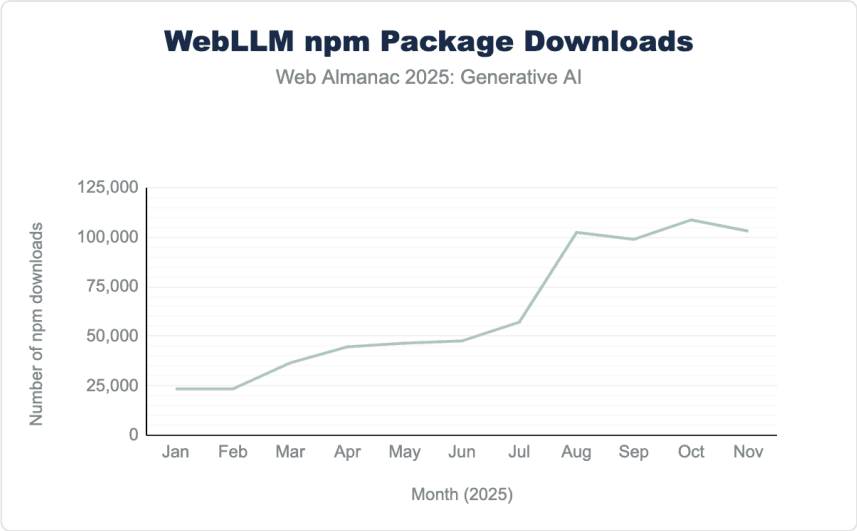


図4.7. `@mlc-ai/web-llm` のnpmパッケージダウンロード数。

WebLLMはブラウザ内LLM推論の最も注目すべきソリューションの1つとして急速に台頭しました。2025年1月から11月にかけて、npmからのWebLLMパッケージのダウンロード数[56]は340%増加しました。8月だけでダウンロード数がほぼ倍増しました。このサージを特定のイベントに帰因することはできませんでした。

Hugging Face[57]が開発したTransformers.js[58]は、人気のPython `transformers` APIを模倣した包括的なJavaScriptライブラリとして機能します。内部ではONNX Runtimeに依存して実行されます。LLMだけでなく、シンプルな高レベルパイプラインを通じてさまざまなタスクの事前学習済みモデルを実行できます。

56. *https://npm-stat.com/charts.html?package=%40mlc-ai%2Fweb-llm&from=2025-01-01&to=2025-11-30*
57. *https://huggingface.co/*
58. *https://huggingface.co/docs/transformers.js/index*

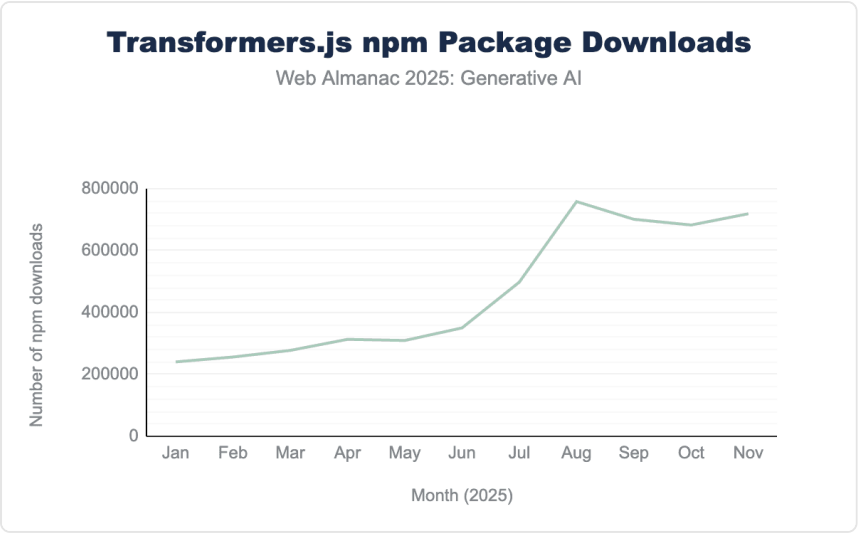


図4.8. `@huggingface/transformers` *のnpmパッケージダウンロード数。*

2025年1月から11月にかけて、npmパッケージのダウンロード数[59]はほぼ3倍になり、2025年8月にも顕著なサージが見られました。

組み込みAI

組み込みAI[60]はGoogleとMicrosoftが推進するイニシアティブで、ウェブ開発者に高レベルのAI APIを提供することを目指しています。BYOAIとは異なり、このアプローチはブラウザ自身が提供するモデルを活用します。開発者は特定のモデルを指定できませんが、この方法によりすべてのウェブサイトが同じモデルを共有でき、一度だけダウンロードすれば良いことを意味します。

このイニシアティブは複数のAPIで構成されています：

- **Prompt API**: 開発者にローカルLLMへの低レベルアクセスを提供します。
- タスク固有のAPI：
    - **Writing Assistance APIs**: テキストの要約、執筆、リライトを行います。
    - **Proofreader API**: テキスト内の誤りを発見・修正します。

59. *https://npm-stat.com/charts.html?package=%40huggingface%2Ftransformers&from=2025-01-01&to=2025-11-30*
60. *https://developer.chrome.com/docs/ai/built-in*

- **Language Detector** と **Translator APIs**: テキストコンテンツの言語を検出し、別の言語に翻訳します。

すべてのAPIはWebMLコミュニティグループ内で仕様策定されており、まだインキュベーション中でW3C勧告トラックには載っていません。一部のAPIはすでに一般提供されていますが、他はブラウザフラグの有効化が必要[61]だったり、Origin Trial[62]中で有効化に登録済みトークンが必要です—安定していて本番機能の出荷に広く利用可能なWebGPUとは異なります。

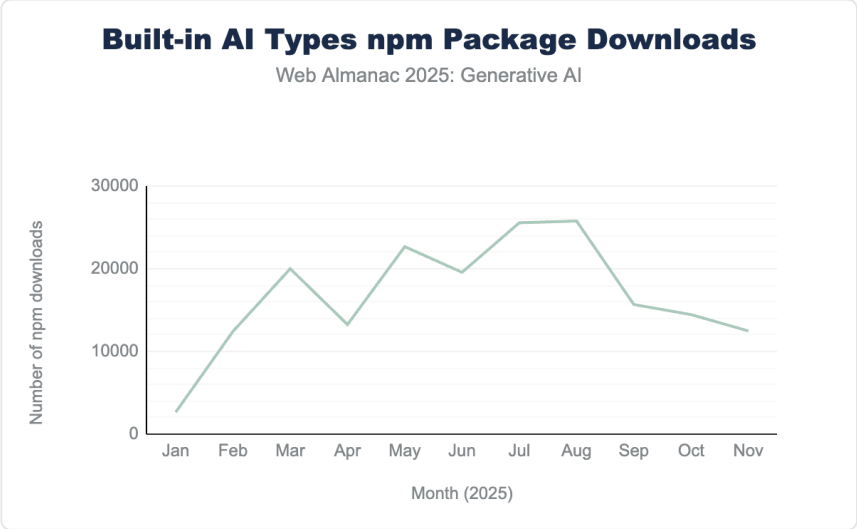


図4.9. `@types/dom-chromium-ai` のnpmパッケージダウンロード数。

組み込みAI向けのTypeScriptタイピングを含む `@types/dom-chromium-ai` パッケージのダウンロード数[63]は、APIの実験的なステータスを反映しているかもしれません：ダウンロード数は2025年8月に25,770でピークに達し、その後緩やかに減少しました。このトレンドは開発者が実験的なAPIの採用に慎重であることを示唆しているかもしれません。ただし、タイピングパッケージのダウンロード統計はAPIの実際の使用状況を反映していない場合があります。

Prompt API

Prompt APIは、ChromeでGemini Nano（Gemini Nano[64]）やEdgeでPhi-4-mini（Phi-4-mini[65]）

61. *https://developer.chrome.com/docs/ai/built-in-apis*
62. *https://developer.chrome.com/docs/web-platform/origin-trials*
63. *https://npm-stat.com/charts.html?package=%40types%2Fdom-chromium-ai&from=2025-01-01&to=2025-11-30*
64. *https://store.google.com/intl/en/ideas/articles/gemini-nano-offline/*
65. *https://learn.microsoft.com/en-us/microsoft-edge/web-platform/prompt-api*

など、ユーザーのブラウザが提供するLLMにアクセスするための標準化されたインターフェースを導入します。これらの一度だけダウンロードするモデルを活用することで、APIはモデルの重みダウンロードを必要とするライブラリに関連する帯域幅の障壁とコールドスタートの遅延を解消します。

しかし、2025年12月時点では技術は移行段階にあります：Chrome 138のブラウザ拡張機能では完全に出荷されています[66]が、ウェブページのアクセスはまだOrigin Trialに制限されています。このAPIは現在モバイルデバイスでは利用できません。

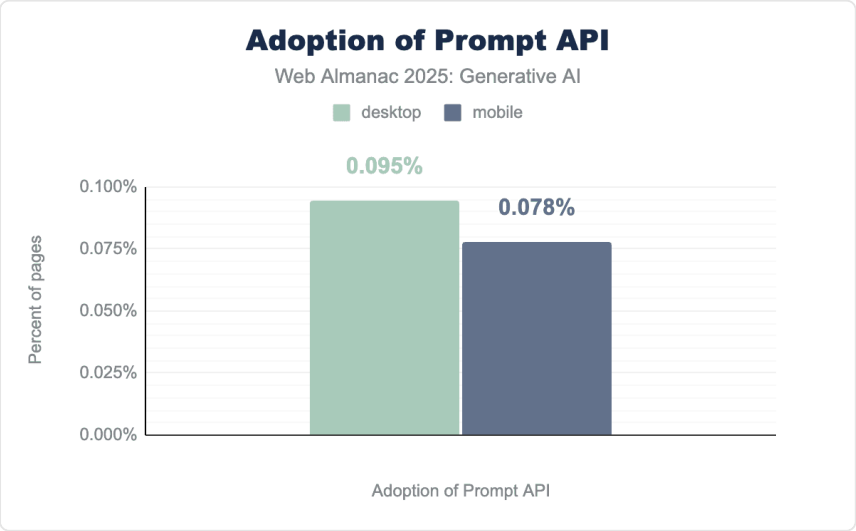


図4.10. Prompt APIの採用状況。

そのため、採用はまだ初期段階にあります。2025年7月（利用可能な最初の計測）のHTTP Archiveデータでは、すべてのデスクトップサイトのわずか0.095%とモバイルサイトの0.078%でAPIが検出され、標準的なユーティリティではなく実験としての現在のステータスを反映しています。

分析したサンプルサイトの多くは、外部スクリプトのGoogle Publisher Tags[67]を通じてPrompt APIを使用していました。このプロジェクトは、著者がウェブサイトに動的な広告を組み込むことを可能にします。Google Publisher Tagsスクリプト[68]はPrompt APIを使用してページのコンテンツをInteractive Advertising Bureau（IAB）コンテンツ分類3.1カテゴリ[69]のリストに分類し、Summarizer API（次のセクションを参照）を使用してページのコンテンツの要約を生成し、両方をサーバーに送信します。ただし、分析中にコードブランチはアクティ

66. *https://developer.chrome.com/docs/ai/prompt-api*
67. *https://developers.google.com/publisher-tag/guides/get-started*
68. *https://securepubads.g.doubleclick.net/pagead/managed/js/gpt/m202512040101/pubads_impl.js*
69. *https://github.com/InteractiveAdvertisingBureau/Taxonomies/blob/develop/Content%20Taxonomies/Content%20Taxonomy%203.1.tsv*

ブではないように見えました。

Writing Assistance APIsとProofreader API

次に、タスク固有のAPIを検討します。Writing Assistance APIs[70]はプロンプトエンジニアリングの複雑さを抽象化します。同じ基盤となる埋め込みLLMを使用しますが、異なる言語的目標を達成するために特殊なシステムプロンプトを適用します：

- **Writer API**: プロンプトに基づいて新しいコンテンツを作成します
- **Rewriter API**: プロンプトに基づいて入力を言い換えます
- **Summarizer API**: テキストの要約を生成します

さらに、Proofreader API[71]は開発者がテキストを校正し、文法エラーやスペルミスを修正することを可能にします。

Summarizer APIはChrome 138で出荷されました。2025年12月時点では、Writer、Rewriter、Proofreader APIはOrigin Trial中です。すべてのAPIは現在モバイルデバイスでは利用できません。

2025年7月のHTTP Archiveクロールでは、`Writer.create()`への呼び出しのみが検出されました（デスクトップサイトの0.127%とモバイルサイトの0.137%）。これは当初Writer APIの使用を示唆していましたが、手動チェックにより、サンプリングされたサイトの多くが実際には同じAPIシグネチャを共有するProtobuf.jsのWriter[72]を使用していることが判明しました。そのため、このメトリクスのチャートは省略することにしました。

Translator APIとLanguage Detector API

組み込みAI APIの最後のカテゴリはTranslator APIとLanguage Detector API[73]で構成されています。Writing Assistance APIsとは異なり、これらのAPIはLLMに依存せず、代わりに特殊なタスク固有のニューラルネットワークを活用しており、生成AIの厳密な定義の外に置かれます。

APIはChrome 138で出荷されました[74]が、現在モバイルデバイスでは利用できません。

70. *https://learn.microsoft.com/en-us/microsoft-edge/web-platform/writing-assistance-apis*
71. *https://developer.chrome.com/docs/ai/proofreader-api*
72. *https://github.com/protobufjs/protobuf.js/blob/827ff8e48253e9041f19ac81168aa046dbdfb041/src/writer.js#L142*
73. *https://developer.mozilla.org/docs/Web/API/Translator_and_Language_Detector_APIs*
74. *https://developer.chrome.com/release-notes/138#web_apis*

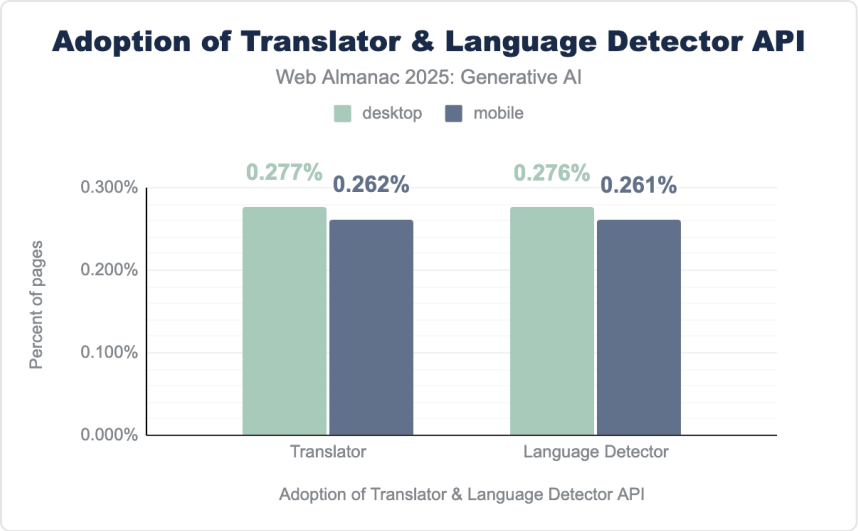


図4.11. Translator APIとLanguage Detector APIの採用状況。

これらは組み込みAI APIの中で最も広い採用を達成しています。2025年7月のHTTP Archiveクロールは Translator APIをすべてのデスクトップサイトの0.277%とモバイルサイトの0.262%で検出し、Language Detector APIはわずか0.001%少ないサイトで使用されていました。

チェックしたサンプルサイトの多くは、Shopifyストアのアドオンとして機能するレビューツールJudge.me[75]を利用していました。Judge.meはLanguage DetectorとTranslator APIの両方を使用しており[76]、これが使用の密な結合の理由かもしれません：Language Detector APIはほぼ同じ絶対数のサイトに存在し、Translator APIにわずか約100サイト遅れていました。

ブラウザ固有のランタイム：Firefox AI Runtime

ChromiumサイドによるAI APIsの代替として、FirefoxはFirefox AI Runtime[77]を試験的に導入しています。これはTransformers.jsとONNX Runtimeをベースにしたネイティブで動作するローカル推論ランタイムです。ただし、このランタイムはまだ一般のウェブからはアクセスできません。拡張機能や、Firefoxの組み込み翻訳機能[78]などの特権的な用途にのみ使用できます。

75. *http://Judge.me*
76. *https://judge.me/help/en/articles/11379816-translating-reviews-in-the-review-widget*
77. *https://firefox-source-docs.mozilla.org/toolkit/components/ml/index.html*
78. *https://www.firefox.com/en-US/features/translate/*

発見可能性

このセクションでは、ウェブにおける生成AIの採用増加に伴うウェブの発見可能性のダイナミクスを見ていきます。AIプラットフォームやサービスがコンテンツを発見する方法に影響を与える2つの重要なアプローチ、`robots.txt`と`llms.txt`ファイルに主に焦点を当てます。

`robots.txt`

ドメインのルートに配置された`robots.txt`ファイル（例：`http://example.com/robots.txt`）は、ウェブサイトオーナーが自動化ボットに対するクロールディレクティブを宣言できるようにします。歴史的にウェブクロールは主に検索エンジンやアーカイブによって行われていました。しかし、生成AIの時代には、ウェブサイトはAIエージェントによっても、またはLLMの学習のためにインターネット規模のデータを収集するモデルプロバイダーによってもクロールされる可能性があります。その結果、ウェブサイトはこれらのクローラーへのアクセスを管理するためにますます`robots.txt`ファイルに依存しています。

OpenAIがモデルのトレーニングに使用する[79]ユーザーエージェント`GPTBot`をターゲットにしたディレクティブの例を以下に示します：

```
User-Agent: GPTBot
Disallow: /
```

`robots.txt`への準拠は任意であり、アクセス制御を厳密に強制するものではないことに注意することが重要です。

79. *https://platform.openai.com/docs/bots*

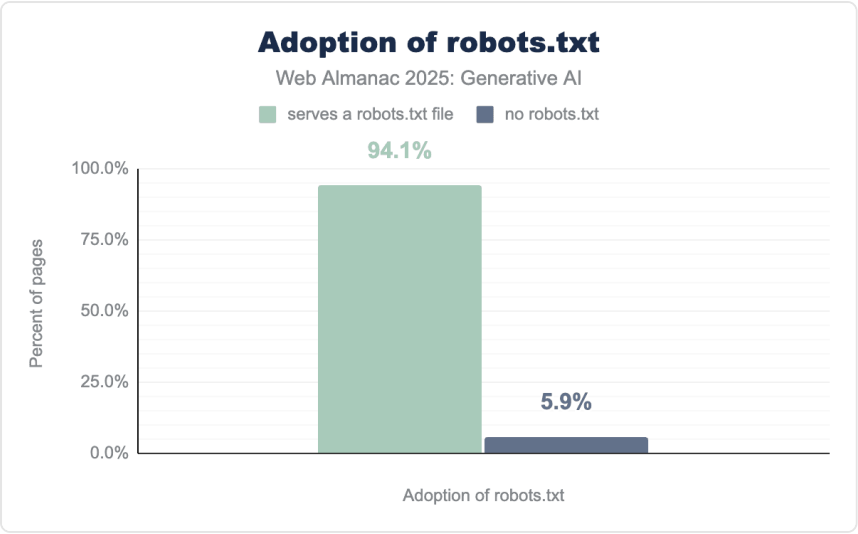


図4.12. `robots.txt` の採用状況。

`robots.txt` の使用は依然として非常に普及しています：分析した約1,290万サイトのうち約94.1%が少なくとも1つのディレクティブを含む `robots.txt` ファイルを持っています。

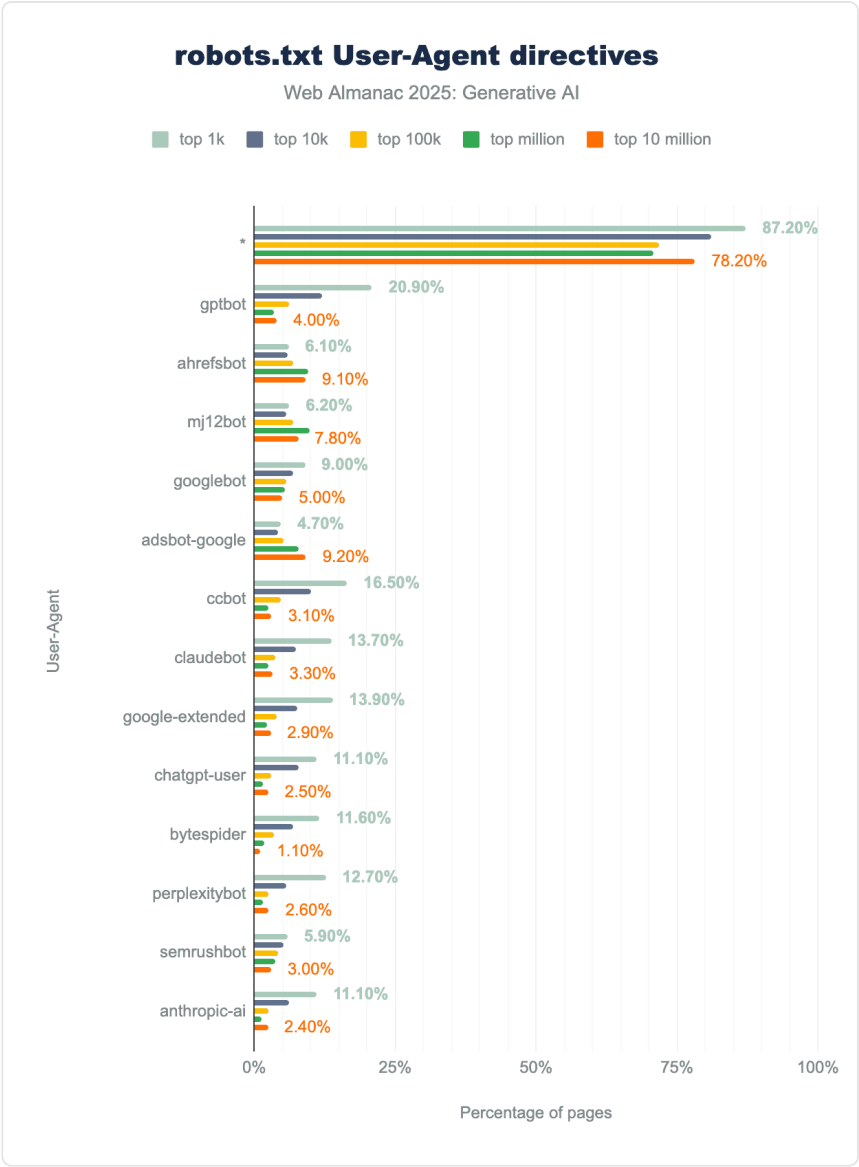


図4.13. ウェブサイトランク別トップ `robots.txt` ディレクティブ。

上図はすべてのウェブサイトで観察されたトップディレクティブを、ランク別にグループ化して示しています。ワイルドカードディレクティブ `*` が最も一般的に使用され、`robots.txt` ファイルの97.4%に存在します。これにより、サイト作者がすべてのボットに対するグローバルなルールを設定できます。2番目に多く使用されるユーザーエージェント

は`gptbot`（分析ではユーザーエージェント名を小文字化したことに注意）で、2024年の2.6%[80]から2025年の約4.5%[81]へと採用が増加しました。

他に頻繁にターゲットにされるAIボットには、Anthropic[82]（`claudebot`、`anthropic-ai`、`claude-web`）、Google（`google-extended`）、OpenAI（`chatgpt-user`）、Perplexity[83]（`perplexitybot`）のものが含まれます。

AIボットをターゲットにしたディレクティブは人気サイトで著しく多く見られ、そのほぼすべてがアクセスを拒否しています。これは人気サイトがオーガニックトラフィックに悪影響を与えることなくAIクローラーを制限できるためと考えられます。全体として、`robots.txt`ファイル内でのAI固有ディレクティブの採用の明確な増加が観察されました。

`llms.txt`

`llms.txt`ファイルは、推論時にLLMがウェブサイトを効果的に活用できるように設計された新興のプロポーザルです。`robots.txt`と同様に、ウェブサイトのドメインのルートにホストされます（例：`https://example.com/llms.txt`）。主な違いはその機能にあります：`robots.txt`はトレーニングや推論中にクロールを許可・禁止するものをボットに指示しますが、`llms.txt`はウェブサイトのコンテンツの構造化されたLLMフレンドリーな概要を提供します。これにより、LLMはユーザーのクエリに答える際にリアルタイムでウェブサイトを効率的にナビゲートできます。

80. *https://almanac.httparchive.org/ja/2024/seo#ai クローラー*
81. *https://almanac.httparchive.org/ja/2025/seo#ai-crawlers-named-in-robotstxt*
82. *https://www.anthropic.com/*
83. *https://www.perplexity.ai/*

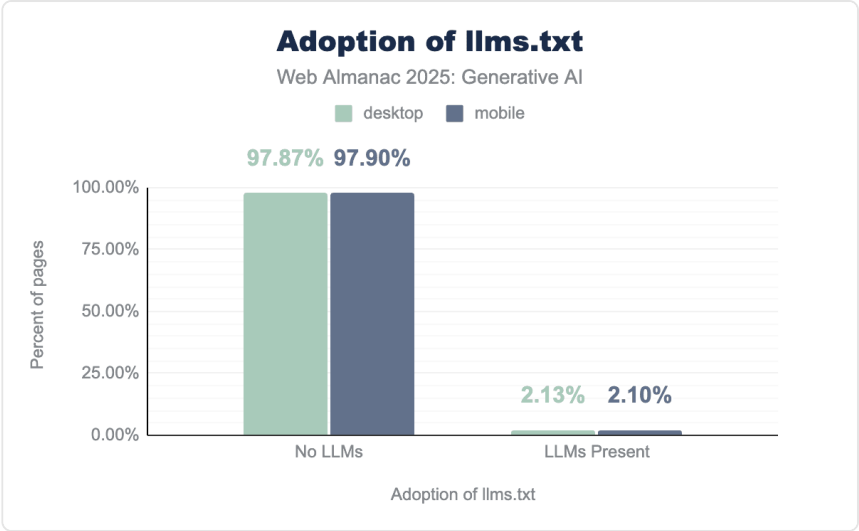


図4.14. `llms.txt` の採用状況。

`llms.txt` は比較的新しいため、ウェブ上での使用は非常に限られていました。デスクトップページでは、有効な `llms.txt` エントリを示すのはわずか2.13%で、97.87%はこのファイルの証拠を示していません。モバイルページも同様のパターンを示し、2.1%のページが有効なエントリを含み、97.9%が含んでいません。

したがって、大多数のサイトはまだ `llms.txt` を通じて明示的なAIアクセス設定を示していません。存在する場合、そのファイルはアーリーアダプターがAI駆動の発見可能性を試験的に実装しているか、エージェントのユースケースを促進しているか、または *生成エンジン最適化*（GEO）を探求していることを示します。検索エンジン最適化（SEO）とは対照的に、GEOはコンテンツをLLMやPerplexityやGoogle AI Mode[84]などのAI対応検索ツールが容易に取り込めるようにすることに焦点を当てています。詳細についてはSEO章を参照してください。

AIフィンガープリント

OpenAI ChatGPT[85]、Google Gemini[86]、Microsoft Copilot[87]などのサービスはすでに広く使用されています。AIの支援を受けてコンテンツが書かれることが増えています。このセクションでは、生成AIがウェブコンテンツとソースコードに与える影響を分析します。

84. *https://search.google/ways-to-search/ai-mode/*
85. *https://chatgpt.com/*
86. *https://gemini.google.com/*
87. *https://www.copilot.com/*

AIカラー

モデルは特定の単語を過剰に使用する傾向があり、それがAIフィンガープリントとして機能します。フロリダ州立大学の研究者たちは2020年から2024年にかけてPubMedに発表された研究論文を比較しました。彼らの分析[88]は、「delves」という単語の使用が驚異の6,697%増加したことを明らかにしました。動詞「delve」の他の活用形も大幅に増加しました。このような指標を個々のケースをAI生成として分類するために使用すべきではありませんが、採用のトレンドを理解するためには依然として有用です。この章のこの部分では、LLMがウェブサイトを生成するために使用する一般的なパターンを掘り下げます（delve）😉。

図4.15. PubMedに発表された研究論文での「delves」という単語の使用増加率。

AI生成ウェブデザインの最も認識しやすい指標は、「AIパープル問題[89]」とも呼ばれる紫色とグラデーションの広範な使用です。

広く採用されているCSSフレームワークTailwind[90]の作成者Adam Wathan[91]は、最近のインタビュー[92]で、このトレンドは主要なAIトレーニングクロールのタイミングから生じている可能性が高いと述べました。2020年代初頭、Tailwindチームは当時のデザイントレンドを反映するためにドキュメントや例に深い紫色を多く使用していました。その結果、そのデータでトレーニングされたLLMは、現在の黒いデザインへの好みなど、より現代的なトレンドに置き換えられた美学を「固定」してしまったようです。

「AIパープル」の美学を定量化するために、ChatGPTのリリース後の年からデータセット内のすべてのルートページのCSS（ウェブサイトのスタイリングを担当）を分析しました。よく知られた`indigo-500`カラー（`#6366f1`）の存在、CSSグラデーションの使用、およびAI生成ウェブサイトに関連する他のカラーを追跡しました。

88. *https://aclanthology.org/2025.coling-main.426.pdf*
89. *https://ai-engineering-trend.medium.com/the-mystery-behind-ais-purple-problem-revealed-0234afdb292e*
90. *https://tailwindcss.com/*
91. *https://x.com/adamwathan*
92. *https://youtu.be/AG_791Y-vs4?t=86*

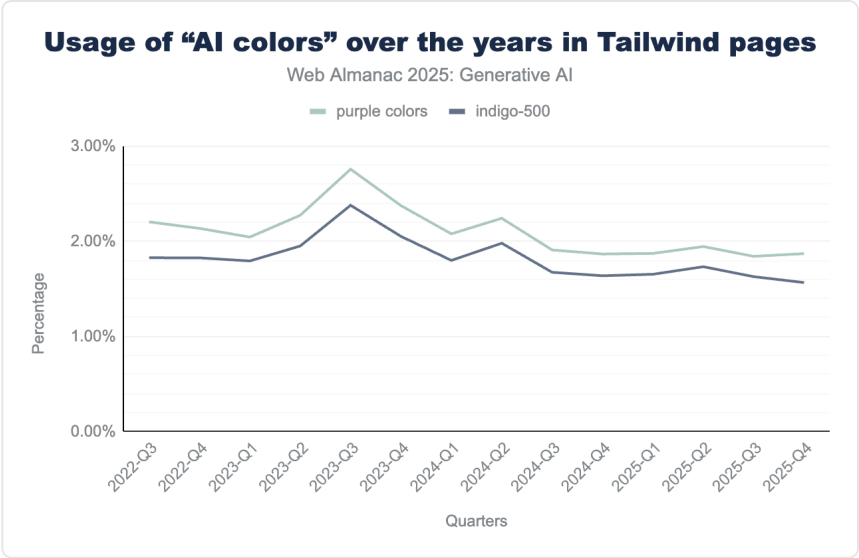


図4.16. Tailwindページにおける「AIカラー」の経年使用状況。

TailwindはClaude、Gemini、OpenAIモデルなどのLLMが好むフレームワークであるため、これらの指標を使用しているTailwindベースのウェブサイトの割合を特に調べました。Tailwindウェブページは HTTP ArchiveのWappalyzerフォーク[93]を使用して識別されました。驚くことに、`indigo-500` または他の2つの一般的に言及されるAIカラー（`#8b5cf6` バイオレットと `#a855f7` パープル）を使用するウェブサイトに顕著なサージは見られませんでした。

93. *https://github.com/HTTPArchive/wappalyzer*

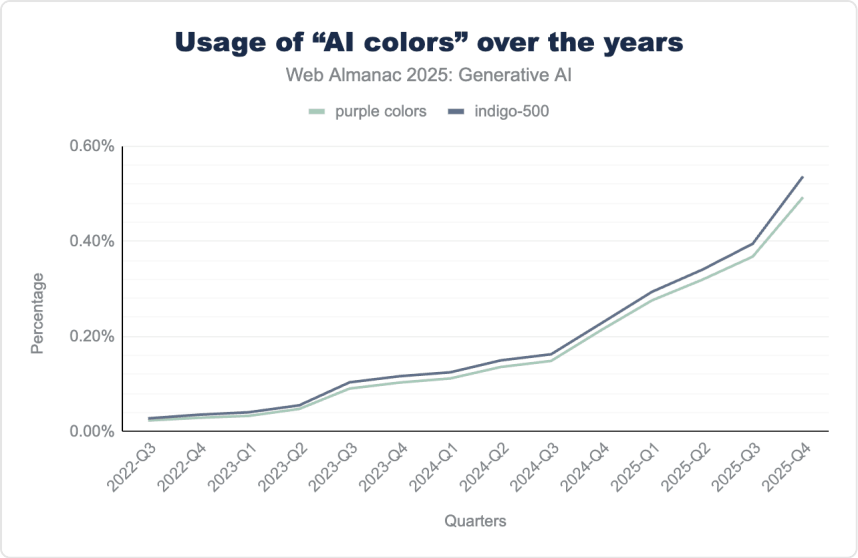


図4.17. 「AIカラー」の経年使用状況。

しかし、Tailwindフレームワークの全体的な爆発的成長により、ウェブ全体の割合として測定した場合、これらの特定のカラーは大幅な増加を示しました。

分析はCSSの変数のみに依存しており、ソースコード全体を解析すると現実的な限界を超えるためです。その結果、私たちの数値は控えめな推定値を表しています。ハードコードされた16進数値（例：`#6366f1`）や古いプリプロセッサを使用しているサイトはクエリで見えませんでした。さらに、カラーの明度を1%調整するだけでも検出を回避できます。

「AIスロップ」[94]や紫がかったデザインに関するオンラインでの頻繁な議論にもかかわらず、HTTP Archiveデータセットの分析は、このトレンドがAIが紫を「選択した」結果というよりも、Tailwindの人気の全般的な上昇を反映しているかもしれないことを示唆しています。

グラデーション、シャドウ、特定のフォントのサージもテストしましたが、統計的に有意な増加は見られませんでした。

バイブコーディングプラットフォーム

新たに作成されるウェブサイトの数が多い理由の1つは、ウェブサイト構築を容易にするプラットフォームの出現です。前の10年間のコンテンツ管理システム（CMS）ツールの普及

94. *https://www.wikipedia.org/wiki/AI_slop*

後、ユーザーはそれらの制約を超えて、v0[95]、Replit[96]、またはLovable[97]などのツールを使用して会話型AIを通じてウェブサイトを生成できるようになりました。これらのプラットフォームでは、独自のドメインまたはプラットフォーム提供のサブドメイン（例：`mywebsite.tool.com`）を使用してページを公開できます。

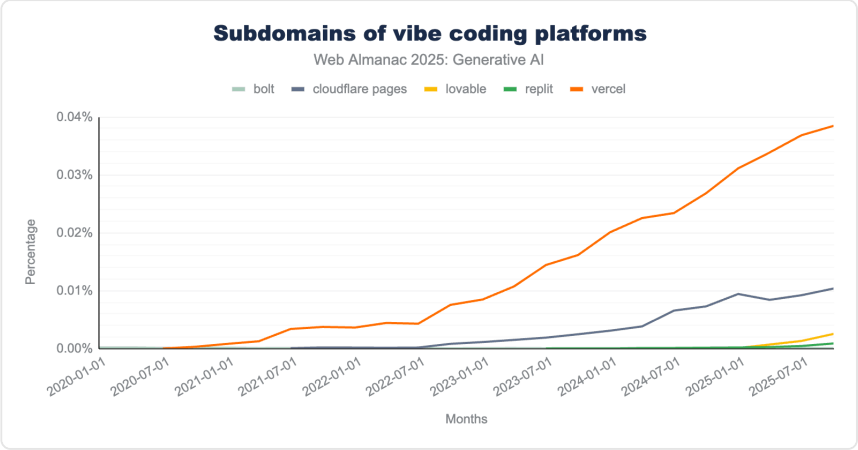


図4.18. バイブコーディングプラットフォームのサブドメイン。

上図はHTTP Archiveデータセット内でのバイブコーディングプラットフォームのサブドメインの相対的な成長を示しています。Vercel（`*.vercel.app`）が依然として支配的なプロバイダーですが、2023年後半のバイブコーディングツールv0（`*.v0.dev`）のリリースよりずっと前からサブドメインホスティングを提供していました。データはv0のローンチ後の即時サージを示していませんでした。Lovable（`*.lovable.app`と`*.lovable.dev`）は2025年初頭の10サブドメインから2025年10月には315に成長しました。これはデータセットで利用可能なページのみを含むことに注意してください。Lovableによると、Lovableでは毎日100,000もの新しいプロジェクトが構築されています[98]。

95. *https://v0.app/*
96. *https://replit.com/*
97. *https://lovable.dev/*
98. *https://lovable.dev/blog/one-year-of-lovable*

`.ai` ドメイン

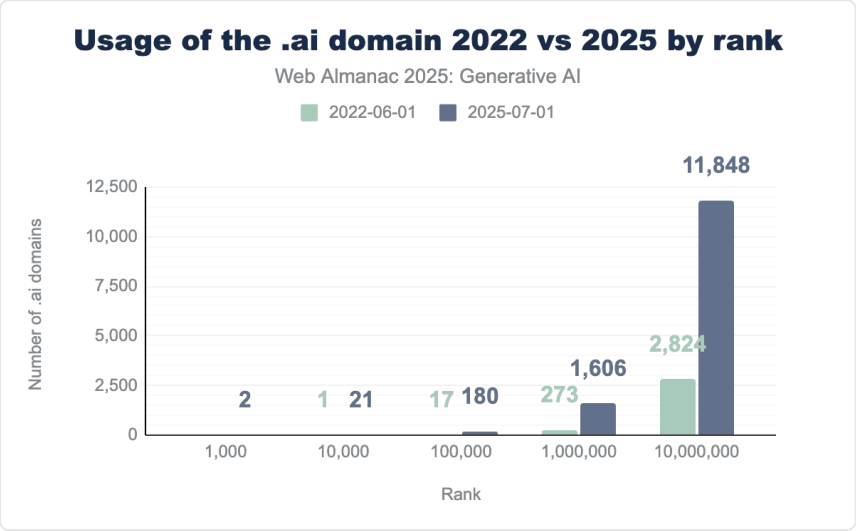


図4.19. ランク別の.aiドメイン使用状況：2022年対2025年。

AIも `.ai` ドメインもChatGPT以前から存在していましたが、ChatGPT登場後に生まれた新たな可能性がAIネイティブビジネスの波をもたらしました。その多くはアンギラの国コード[99]である `.ai` トップレベルドメインを採用しました。このトレンドは、上図に示す2022年のChatGPT以前の状況と2025年の使用状況の比較が示すように、すべてのランクにわたって `.ai` ドメイン登録の大幅な増加をもたらしました。

Chromeデータセットでトップ1,000の最も訪問されたウェブサイトに入った `.ai` ドメインは、https://character.aiとアダルトサイトの2つのみでした。

結論

2025年、生成AIはクラウド専用技術から基本的なブラウザコンポーネントへと移行しました。BYOAIはWebAssemblyやWebGPUなどの基盤APIの成長、またWebLLMやTransformers.jsなどのライブラリに後押しされて即時採用を支配しました。同時に、組み込みAIはブラウザ内に直接標準化されたAIレイヤーの基盤を築き始めました。より制限的な `robots.txt` ファイルと歓迎的な `llms.txt` の構造の間に緊張関係が生まれています。バイブコーディングと `.ai` ドメインの台頭は、AIがアプリの機能だけでなく、アプリの構築方法も再形成していることを証明しています。

99. *https://www.iana.org/domains/root/db/ai.html*

2025年を超えて見ると、次の進化の飛躍はすでに見えています：エージェントAIです。インタラクティブなチャットボットの時代から、継続的なユーザー介入なしに意思決定、マルチステップワークフローの実行、複雑なタスクの解決ができる自律エージェントへと移行しています。このシフトは、Perplexity Comet[100]、ChatGPT Atlas[101]、またはOpera Neon[102]などの新しいクラスの「エージェントブラウザ」の台頭をもたらしています。

これらのエージェントをウェブの広大なリソースに接続するために、WebMCP[103]（Web Model Context Protocol）などの新しいプロトコルが登場しています。HTTPがリソースの送信を標準化するように、WebMCPはエージェントがウェブインターフェースを認識し操作する方法を標準化することを目指しており、AIネイティブウェブが読み取り可能なだけでなく、アクション可能であることを保証します。

著者

Christian Liebel

@christianliebel @https://mastodon.cloud/@christianliebel christianliebel

christianliebel https://christianliebel.com

Christian Liebel（修士）は、W3C技術アーキテクチャグループ[104]（TAG）の選出メンバー、W3C Web Machine Learning[105]（WebML）コミュニティグループおよびワーキンググループの参加者、そしてWeb AIのGoogle Developer Expert（GDE）です。

Yash Vekaria

@vekariayash Yash-Vekaria https://www.yashvekaria.com/

Yash Vekariaはカリフォルニア大学デービス校[106]のコンピュータサイエンス博士候補者です。ウェブベースの大規模インターネット計測を行い、ウェブのダイナミクスの研究と改善に取り組んでいます。特に、オンライントラッキング慣行とユーザープライバシー問題の研究と透明性向上に焦点を当てています。

100. *https://www.perplexity.ai/comet*
101. *https://chatgpt.com/atlas/*
102. *https://www.operaneon.com/*
103. *https://github.com/webmachinelearning/webmcp*
104. *https://w3ctag.org/*
105. *https://webmachinelearning.github.io/*
106. *https://www.ucdavis.edu/*

Jonathan Pagel

jcmpagel jonathan-pagel https://jonathanpagel.com

Jonathan Pagelは学士課程でeコマースを学び、以来この分野、特にショップやウェブサイトの速度最適化とアクセシビリティに関心を持っています。現在はこの分野でフリーランスとして活動しながら、AI・社会論の修士号取得を目指しています。

部 II 章 5
SEO

Amaka Chulwuma、 Chris Green と Sophie Brannon によって書かれた。
Jamie Indigo によってレビュ。
Augustin Delport と Chris Green による分析。
Michael Lewittes、 Montserrat Cano と Sharon McClintic 編集。
Sakae Kotaro によって翻訳された。

はじめに

検索エンジン最適化（SEO）は、オンラインで情報がどのように発見・理解されるかにおいて引き続き中心的な役割を担っています。SEOはウェブサイトが効果的にクロールされ、インデックスされ、検索結果に表示されるかどうかを決定する技術的・構造的・コンテンツ的な施策全般を包含します。

強固なSEOの基盤は、従来の検索エンジンでの視認性を支えるだけでなく、AIシステムがウェブコンテンツを新しい方法で解釈・要約し始めるにつれ、その重要性はますます高まっています。

2025年版Web AlmanacのSEOチャプターは、HTTP Archiveのクロール、Lighthouseレポート、Chrome User Experience（CrUX）レポート、およびカスタム指標からデータとインサイトを収集しています。ウェブの技術的な状態がどのように進化しているかを記録し、今日のオーガニック視認性に最も影響を与える要因を特定することが目標です。

多くのSEO指標がウェブ全体で安定している一方、それを取り巻く状況は急速に変化しています。AIクローラーの台頭、`llms.txt` の登場、機械可読性への関心の高まりは、最適化がもはやボットに *見つけてもらう* だけでなく、ボットに *理解してもらう* ことについてであることを示唆しています。オンライン検索のこうした変化するコンテキストは、今後の最適化にどのような影響を与えるのでしょうか。また、最新のデータポイントはすでにAIのSEOへの影響をどのように反映しているのでしょうか？

クロール可能性とインデックス可能性

ウェブコンテンツが検索結果で視認されるためには、まず検索エンジンのクローラーにクロールされ、インデックスされる必要があります。クロール可能性はボットがページを見つけてアクセスできるかどうかを決定し、インデックス可能性はそのページが検索結果に表示される資格があるかどうかを定義します。この2つの概念が合わさって、検索の視認性の基本要素を形成します。ボットに最初に発見・理解されなければ、ページはランク付けされたりユーザーに提供されたりすることができません。同様に、サイトがインデックス可能でなければ、コンテンツはAIプラットフォームで引用されません。

Googleのドキュメント[107]は、クロールルールの遵守が `robots.txt` ファイルの存在だけでなく、そのファイルが正しく構成されているかどうかにも依存することを明確にしています。検索エンジンはまた、ディレクティブを効率的に解析し、一貫して解釈できるよう、実際の制限と標準化されたキャッシュ動作を適用しています。

同時に、Cloudflareが説明するように[108]、`robots.txt` は常に遵守されるコマンドというよりも、行動規範として機能します。評判の良いボットはこれらのシグナルを尊重しますが、他のボットはまったく無視する場合もあります。この協調と予測不可能性の混在が現代のクロール環境を定義し、サイトが実際にクローラーアクセスをどのように管理しているかを検討する舞台を設定しています。

`robots.txt`

クローラーのためのウェブの事実上の「案内所」として機能する `robots.txt` ファイルは、ボットがサイトのどの部分が公開されているか、または制限されているかを確認する場所です。3年前にIETFがRobots Exclusion Protocol（RFC 9309[109]）を標準化[110]して以来、その構文、キャッシュ動作、エラー処理が明確に定義され、クローラーがアクセスルールを解釈するための安定したフレームワークが提供されています。

そのフレームワークを改良する取り組みは継続中です。2024年後半、IETFはREPextという

107. *https://developers.google.com/search/docs/crawling-indexing/robots/robots_txt*
108. *https://www.cloudflare.com/learning/bots/what-is-robots-txt/*
109. *https://datatracker.ietf.org/doc/html/rfc9309*
110. *https://www.ietf.org/about/*

ワーキングドラフトを導入しました[111]。これはRFC 9309を基に、レスポンスヘッダーとHTML `meta` タグを通じたページレベルのクロール制御を探求するもので、将来の実装をより細かく柔軟にする可能性があります。

しかし現時点では、`robots.txt` ファイルはクロール管理の基盤であり続けています。ほとんどのウェブサイトは有効なファイルを提供しており、省略しているのはごく少数です。省略しているサイトの場合、サイト所有者は複雑なボット固有のルールよりも、シンプルで普遍的なディレクティブを好む傾向があります。続くセクションでは、これらの傾向が2025年のデータにどのように現れているかを検討します。

`robots.txt` ステータスコード

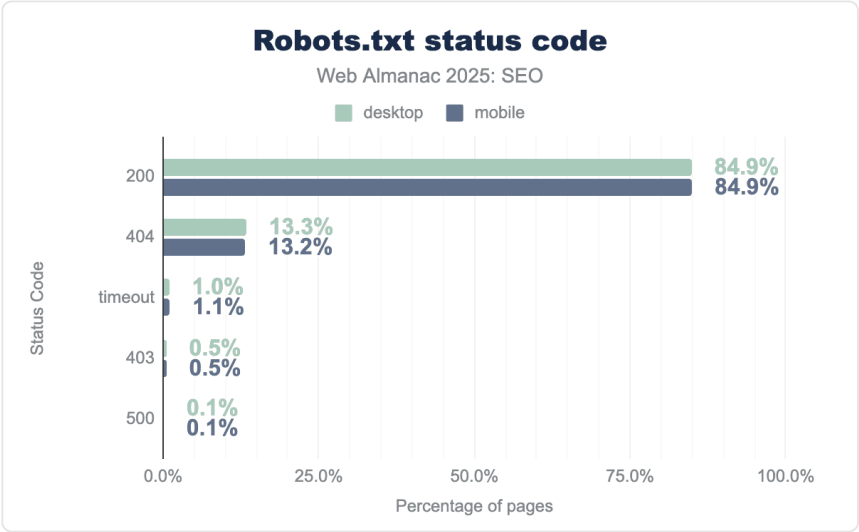


図5.1. `robots.txt` のステータスコード。

2025年には、`robots.txt` ファイルへのリクエストの85%が有効な200ステータスコードを返しており、2024年の84%から増加しています。この若干の上昇は、2022年の標準化の波及効果と、有効な `robots.txt` ファイルを提供するCMSのデフォルト設定の普及が続いていることを示唆しています。ただし重要なのは、200レスポンスはファイルが存在することを確認するだけで、そのディレクティブが正しいまたはサイトにとって有益であることを保証するものではありません。

有効な（200）`robots.txt` ステータスコードに関しては、モバイルとデスクトップの差はほぼ消失しており、現在モバイルはデスクトップよりわずか0.06%リードしています。これ

111. *https://garyillyes.github.io/ietf-rep-ext/draft-illyes-repext.html*

は業界が独立した「mドット」サイトから統一された設定のレスポンシブデザインへ移行していることを反映しています。

`robots.txt` ファイルに対する404エラーの割合は2024年の14%から13%に低下しました。見つからないファイルの減少は、クロール無制限をデフォルトとするのではなく、より多くのサイトが明示的に `robots.txt` ファイルを提供していることを意味します。

タイムアウトは約1.0%、403レスポンスは約0.5%、5xxは約0.1%です。まれではありますが、`robots.txt` ファイルへの5xxレスポンス[112]は、キャッシュされたファイルまたは後続の成功したフェッチが利用可能になるまで、検索エンジンがサイトを一時的にブロック済みとして扱う原因になりえます。

`robots.txt` ファイルサイズ

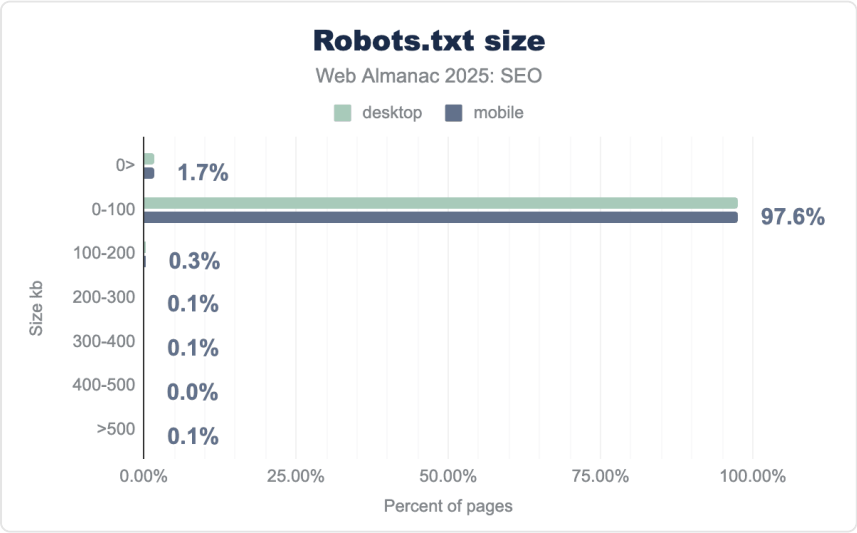


図5.2. `robots.txt` のサイズ。

ほぼすべての `robots.txt` ファイルはサイズ制限（Googleは500KBの[113]解析カットオフを強制）を大幅に下回っており、空のファイルを提供しないなどの標準に準拠しています。

ごく少数のサイトが完全に空の `robots.txt` ファイルを提供しており、現在デスクトップで1.8%、モバイルで1.7%と2024年からわずかに増加しています。ほとんどの主要クローラ

112. https://developers.google.com/search/docs/crawling-indexing/robots/robots_txt#:~:text=Other%20errors-,A%20robots.,treated%20as%20a%20server%20error.
113. https://developers.google.com/search/docs/crawling-indexing/robots/robots_txt#:~:text=Google%20enforces%20a%20robots.,the%20size%20of%20the%20robots.

ーは空のファイルを許可的に扱います[114]が、標準（RFC 9309[115]）はこの動作を明示的に定義していないため、あまり知られていないボットによる一貫性のない処理の余地が残ります。制限が意図されていない場合は、有効なファイルまたは404を返すのがより安全なアプローチです。

2025年には、ファイルの98%が100KB未満であり、2024年からわずかに低下しています。年間のこの小さな変化は、安定した成熟した実装パターンを示しています。

100〜200KBのファイルは `robots.txt` の0.3%、200〜300KBはわずか0.1%を占め、前年からほぼ変化がありません。Googleが強制する500KBの解析カットオフを超えたサイトはわずか0.1%で、クローラー制限への強い準拠を確認しており、過大サイズの `robots.txt` ファイルはエッジケースであることを裏付けています。

`robots.txt` ファイルサイズがクロール可能性の障壁になることはほとんどありません。より差し迫った問題は、空または設定ミスのファイルで、クローラーがサイトルールを解釈する際に不確実性をもたらします。

`robots.txt` ユーザーエージェントの使用状況

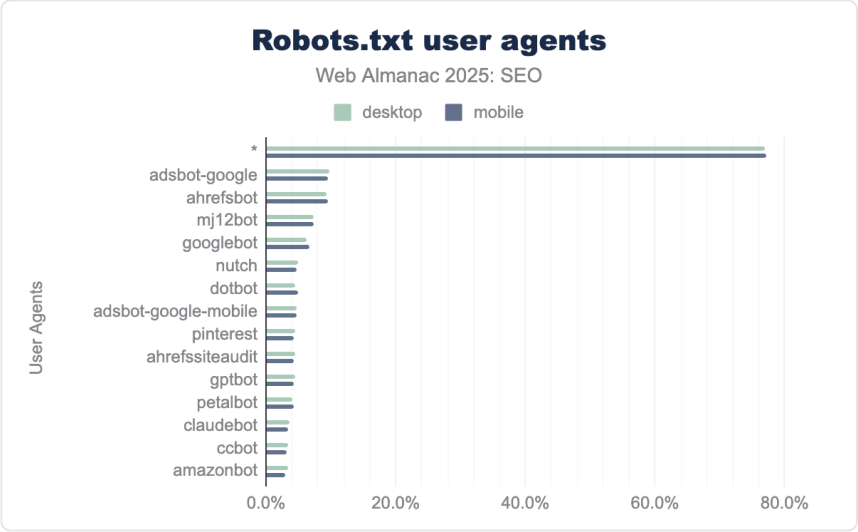


図5.3. `robots.txt` のユーザーエージェント。

分析のため、クロールで識別されたすべてのユーザーエージェント名を小文字化し、サイト

114. *https://developers.google.com/search/docs/crawling-indexing/robots/robots_txt#:~:text=For%20the%20first%2012%20hours,checking%20for%20a%20new%20version).*
115. *https://www.rfc-editor.org/rfc/rfc9309.html*

間の差異を無視しました。一部のクローラーはドキュメントで推奨する大文字化を提供していますが、`robots.txt` ファイルを処理する際には理論上、大文字と小文字を区別しないマッチングを使用する必要があります[116]。

キャッチオールユーザーエージェント `*` は、今日の `robots.txt` ファイルで見られるクローラーディレクティブとして最も一般的なアプローチです。2025年には77%のファイルに表示されており、2024年から、また2022年の70%台半ばから若干増加しています。この着実な上昇は、ほとんどのサイト所有者が複雑なボット固有の指示を維持するよりも、広範な普遍的ルールを実装することを好むことを示しています。

`robots.txt` ファイルで特定のユーザーエージェント が言及されている場合、Googleの広告クローラー（`adsbot-google`）と `ahrefsbots` が昨年と同様にリードしています。

`robots.txt` ディレクティブにおける `adsbot-google` のターゲティングは2025年にデスクトップで9.8%、モバイルで9.5%に上昇しており、2024年のデスクトップ9.1%、モバイル8.9%と比較しています。

`ahrefsbot` も主要な名前付きクローラーであり続け、デスクトップの9.3%、モバイルの9.5%の `robots.txt` ファイルで指定されています。`ahrefssiteaudit`（デスクトップ4.6%/モバイル4.3%）と合わせると、大量のクロール活動を生み出しうるSEOツールのアクセスを制御することの継続的な重要性を反映しています。

今年注目すべき量で表示されたその他の名前付きクローラーには次のものがあります：

- `mj12bot`（Majestic）: デスクトップ7.3% / モバイル7.3%
- `googlebot`: デスクトップ6.2% / モバイル6.7%
- `nutch`: デスクトップ5.0% / モバイル4.8%

`bingbot` は `robots.txt` でほとんど名指しされない

`bingbot` は `robots.txt` ファイルの名前付きユーザーエージェントの中で22位にランクし、`robots.txt` の3%未満に表示されます。ボットが `robots.txt` ファイルに表示される場合、ウェブサイト管理者がそのクローラーを許可、制限、またはクロール率の設定など、その動作を明示的に制御するほど関心を持っていることを意味します。低い表示率は無視されていることを示唆しています。MicrosoftのAIへの大規模な投資[117]とBingへのChatGPTの統合[118]にもかかわらず、クローラー自体は `robots.txt` ファイルでより目立つ存在にはなっていません。AI機能強化にもかかわらず、Bingのウェブフットプリントとサイトオペレー

116. *https://datatracker.ietf.org/doc/html/rfc9309#section-2.2.1:~:text=Crawlers%20MUST%20use%20case%2Dinsensitive%20matching%20to%20find%20the%20group%20that%20matches%20the%20product%20token%20and%2*
117. *https://blogs.microsoft.com/on-the-issues/2025/01/03/the-golden-opportunity-for-american-ai/*
118. *https://blogs.microsoft.com/blog/2023/02/07/reinventing-search-with-a-new-ai-powered-microsoft-bing-and-edge-your-copilot-for-the-web/*

ターにとっての重要性はほぼ変わらないままで、`robots.txt` ファイルで名指しされる `gptbot` のようなAI特化クローラーの急速な台頭との静かな対照をなしています（詳細は後述）。

名前付きユーザーエージェント使用の緩やかな増加

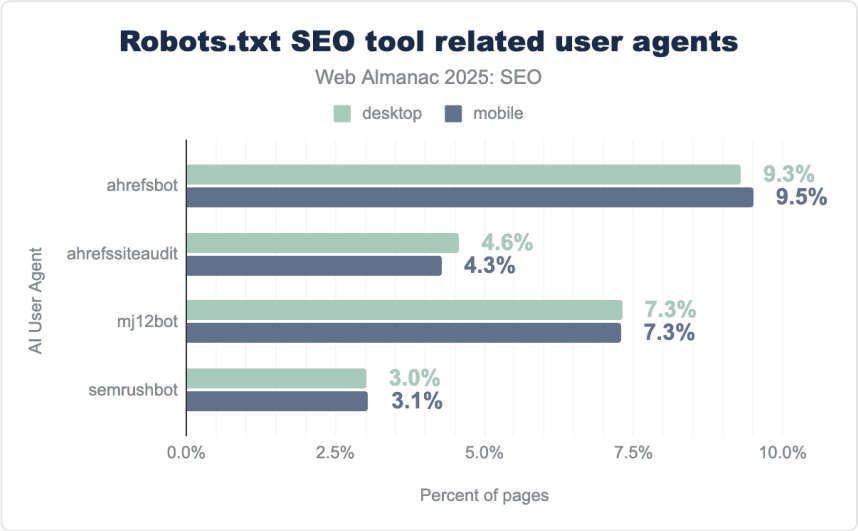


図5.4. `robots.txt` のSEOツール関連ユーザーエージェント。

全体として、2024年と比較して、2025年の `robots.txt` ファイルでは、普遍的なワイルドカード `*` のみに頼るのではなく、名前付きクローラーのディレクティブを含む割合がわずかに大きくなっています。たとえば、MJ12Botは昨年のモバイル6.6%から2025年のモバイル7.3%に上昇し、Googlebotはモバイル6.4%から6.7%に、Nutchはモバイル4.3%から4.81%に増加しました。これらの控えめな伸びは、普遍的な制御のシンプルさから離れることなく、重要な箇所でカスタマイズされたクロールルールを設定するサイト所有者が増えているという、段階的な改善を示しています。

特定のボットへの言及が増加する中での `*` の継続的な優位性は、実用的なバランスを示唆しています。普遍的なディレクティブが標準のままですが、ビジネス上の懸念がある箇所には対象を絞ったルールが追加されます。すべてのクローラーが `*` を一貫して解釈するわけではありません。GoogleのAdsBotはそれを無視し、Applebotは `*` を適用する前にGooglebotルールにフォールバックするため、特定のケースでは明示的なターゲティングが必要です。

`robots.txt` でAIクローラーを名指し

AIクローラーの台頭により、`robots.txt` は従来の検索最適化ツールからコンテンツ権限を管理するための広範なメカニズムへと変貌しました。2025年には、AI関連ボットをターゲットにしたディレクティブが2024年の水準から大幅に増加しました。

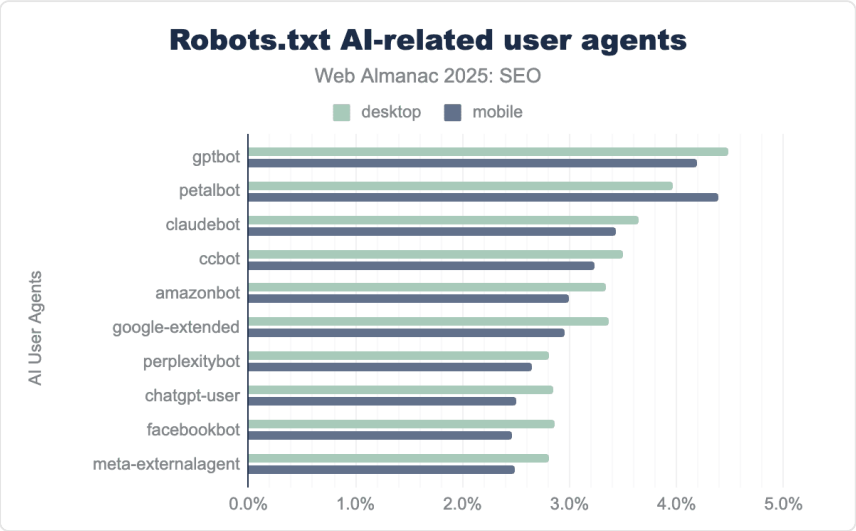


図5.5. `robots.txt` のAI関連ユーザーエージェント。

前年比の比較は、採用の加速を明らかにしています：

- `gptbot`: デスクトップ4.5%、モバイル4.2%と、2024年のデスクトップ2.9%、モバイル2.7%から約55%増加。
- `ccbot`: デスクトップ3.5%、モバイル3.2%と、2024年のデスクトップ2.7%、モバイル2.4%から増加。
- `petalbot`: 今年デスクトップ4.0%、モバイル4.4%；2024年には個別に追跡されていなかった。
- `claudebot`: デスクトップ3.6%、モバイル3.4%と、2024年のデスクトップ1.9%、モバイル1.6%からほぼ2倍。

注目すべきは、2024年のデータには `anthropic-ai`（昨年デスクトップ2.0%、モバイル1.7%）や `chatgpt-user`（昨年デスクトップ2.0%、モバイル1.7%）などの広範なカテゴリーが含まれていたのに対し、2025年のデータはより特定のボットターゲティングを示しています。

今年AIクローラー分野に参入したその他のボットには、`amazonbot`（デスクトップ3.3%、モバイル3.0%）、`google-extended`（デスクトップ3.4%、モバイル3.0%）、`perplexitybot`（デスクトップ2.8%、モバイル2.7%）、`chatgpt-user`（デスクトップ2.8%、モバイル2.5%）、`facebookbot`（デスクトップ2.9%、モバイル2.5%）、`meta-externalagent`（デスクトップ2.8%、モバイル2.5%）があります。

ユーザーエージェントのターゲティング全体は年々緩やかに成長しているのに対し、AIクローラーの採用ははるかに急激です。2024年以降の `gptbot` のほぼ倍増、`petalbot`、`claudebot` などの測定可能な採用は、2023年のわずかな存在から2025年には数パーセントの採用率に達した、名前付きユーザーエージェントに対する `robots.txt` ディレクティブの最近の記憶の中で最も急速な拡張の一つを表しています。

この変化はサイト所有者に新たな複雑さをもたらしています。「このページは検索にインデックスされるべきか？」と問うだけでなく、「このコンテンツはAIモデルのトレーニングに使用されるべきか？」とも問わなければなりません。これらは異なるビジネス的・倫理的含意を持つ別々の考慮事項です。

`robots.txt` は、検索での視認性とデータの大規模収集からの保護のバランスを取る二重目的の制御ポイントになっています。AIモデルとクローラーが増殖するにつれ、このトレンドは激化する可能性があります。

`llms.txt`

`llms.txt` ファイルは、「推論時にLLMがウェブサイトを使用するのに役立つ情報を提供する」新しい標準として提案されています（llmstxt.org[119]より）。このテキストファイルはMarkdown形式のウェブサイトコンテンツの高度に簡略化されたバージョンを含み、LLMが生成レスポンスで取り込んで使用しやすくすることを目指しています。

この標準はまだ広く採用されておらず、SEO業界全体で議論の的となっています。Googleは `llms.txt` を使用していないとしばしば述べており、現在Googleのサービスは使用していません[120]。しかし、Anthropicは `llms.txt` のリードを取っており、このフォーマットがモデル推論中のコンテンツ利用を管理・最適化するための信頼できるメカニズムへと発展するかもしれないという楽観論が高まっています。

`llms.txt` の採用率

`llms.txt` の使用がSEOやAI最適化に有効なアプローチであることが証明されるかどうかに

119. *http://llmstxt.org*
120. *https://www.seroundtable.com/google-ai-llms-txt-39607.html*

かかわらず、このような新しいファイル形式と提案された標準の導入は、今後のウェブサイトの構築と最適化に影響を与える可能性があります。

2025年Almanacの一環として、ウェブ全体の `llms.txt` 採用レベルを評価するモニターを導入しました。

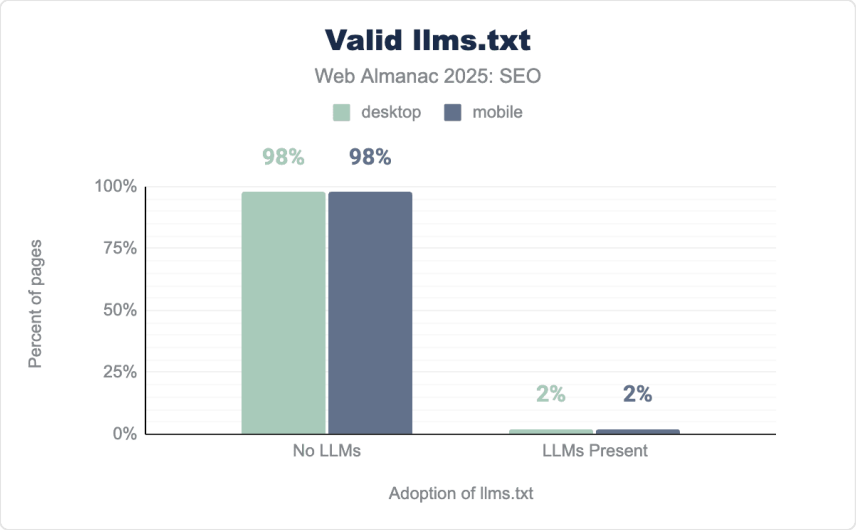


図5.6. 有効な `llms.txt`

採用率は2%と比較的低いですが、`llms.txt` 形式がどれほど新しいか、および分析されたすべてのモバイルサイトで合計324,184の有効なファイルが確認されたことを考えると、これらの数値は依然として注目に値します。

これらの `llms.txt` ファイルの内容を掘り下げると、これまでのこの標準の採用について、そして将来この動きを促すものについていくつかの手がかりが見えます。

- `llms.txt` ファイルの39.6%はAll in One SEOに関連
- `llms.txt` ファイルの3.6%はYoast SEOに関連

これはこれらのCMS拡張が生成したファイルに（デフォルトで）残すコメントから得られましたが、`llms.txt` ファイルを持つウェブサイト所有者の相当数がCMS/アドオン拡張によって生成されていることを示しています。したがって、これが常に意識的な行為または `llms.txt` 標準の支持なのか、意図せぬ組み込みなのかは確認できません。

2025年1月以来、この新興標準への関心は高まり続けており[121]、1〜2社の主要なAI企業の認知にかかっています。それでも、2026年の数字を見るのは興味深いでしょう。

ロボットディレクティブ

ロボットディレクティブ[122]は、特定のページがどのようにインデックスされ検索結果に表示されるかに対するページレベルの制御を提供します。`robots.txt` ファイルと同様の機能を持ちますが、両者は異なる目的を持ちます：

- ロボット *ディレクティブ* は **インデックスとサービング** に影響する
- 一方 `robots.txt` は **クロール** を制御する

ディレクティブが適用されるためには、クローラーがページにアクセスできる必要があります。ページが robots.txt によってブロックされている場合、そのディレクティブが見られたり従われたりすることはないかもしれません。

ロボットディレクティブの実装

ロボットディレクティブを実装する主な方法は2つあります：

1. `<meta name=robots>` タグを使用する（ウェブページの `<head>` セクション内に配置）
2. `X-Robots-Tag` HTTPヘッダーを使用する

どちらの方法を選ぶかは、具体的なユースケースと利用可能な手段や方法によって異なります。

`meta` ロボットタグは主にHTMLページ向けで、ほとんどの主要なCMSでネイティブまたはアドオンを通じて広くサポートされており、その他の一般的でよくサポートされているフレームワークでも同様です。

`X-Robots-Tags` はPDFや文書などの他のHTML以外のファイルタイプにも実装できるという大きな利点があります。ただし、CMSユーザーにとっては設定が簡単でないことが多く、常に選択肢として使えるわけではありません。

121. *https://trends.google.com/trends/explore?q=llms.txt*
122. *https://developers.google.com/search/docs/crawling-indexing/robots-meta-tag*

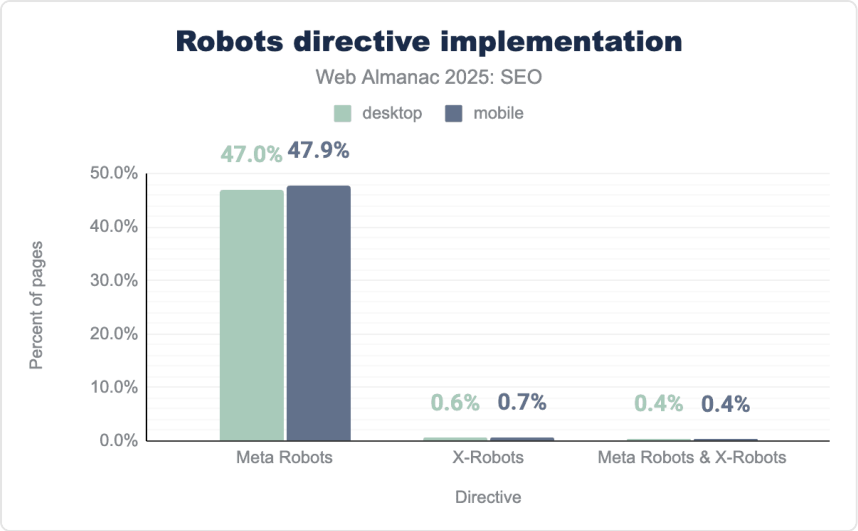


図5.7. ロボットディレクティブの実装

`meta` ロボットはロボットディレクティブを実装するための最も広く使われている方法で、デスクトップページの47.0%、モバイルページの47.9%に表示されており、X-Robotsはデスクトップ0.6%、モバイル0.7%で大差で2番手につけています。

`meta` ロボットを実装しているページ数は昨年記録されたデスクトップ45.5%、モバイル46.2%から増加しており、前年比（YoY）の成長を示しています。対照的に、`X-Robots-Tag` の実装はYoYでほぼ変わっていません。インナーページはロボットディレクティブを持つ可能性が50%で、ホームページの46%と比較しています。

一部のページ（デスクトップとモバイルの両方で0.4%）では `meta` ロボットと `X-Robot-Tag` の両方が同時に実装されています。この0.4%という数字は昨年から安定していますが、2つの実装方法間で矛盾するシグナルが生成される可能性が高まるため、広く推奨されている慣行ではありません。

ロボットディレクティブのルール

実装方法はロボットディレクティブの一側面に過ぎません。ディレクティブルール[123]がページやドキュメントをどのように処理するかを決定します。

ルールはコンマ区切りの値として `meta` ロボットまたは `X-Robots-Tag` のいずれかに追加できます。

123. *https://developers.google.com/search/docs/crawling-indexing/robots-meta-tag#directives*

ディレクティブルールの調査では、レンダリングされたHTMLに依存しました。

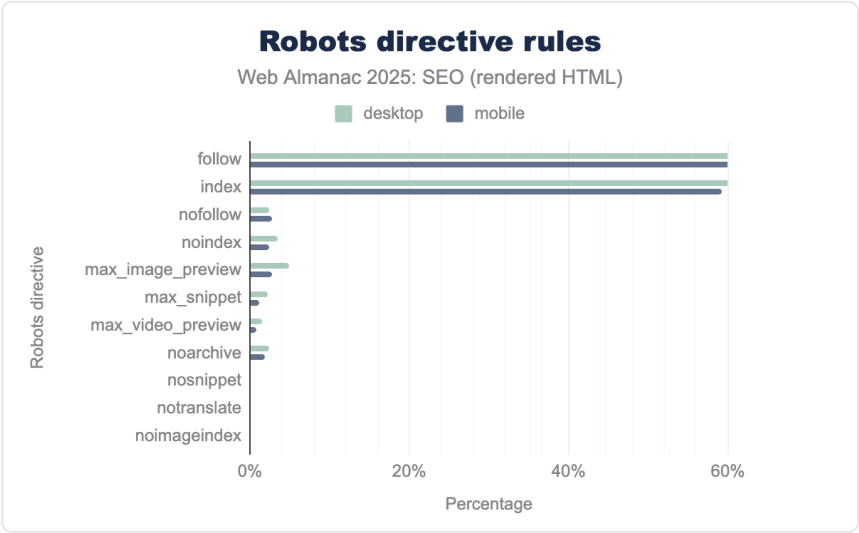


図5.8. ロボットディレクティブのルール

最もよく使われる2つのルール `follow` と `index` は、`noindex` や `nofollow` などの反対の目的を持つ他のルールがない場合のデフォルト値とみなされ（Googleには無視されます）。

これらを含めることは、ロボットがページをインデックスし、そこからリンクをたどるべきであることを意味します。モバイルでの `follow` と `index` の使用率はそれぞれ60.5%と59.3%であり、これらの2つのタグが一緒に見られ、一般的に補完的であることを意味しています。

技術的に不要な `meta` ロボットルールの組み合わせがこれほど多い可能性のある原因のひとつはYoast SEOで、`index,follow` をデフォルトで適用します。Yoastはホームページでのをてのセオツール/プラグインの使用を見ると約16%の採用率（デスクトップとモバイル）を持ち、識別されたすべてのSEOツール[124]のうち、70%近くの時間で使用されています。

`nofollow` と `noindex` は次に最もよく使われる `meta` ロボットルールで、使用頻度はかなり低く、デスクトップページの2.8%、モバイルページの2.4%に表示されます。

124. *https://www.wappalyzer.com/technologies/seo/*

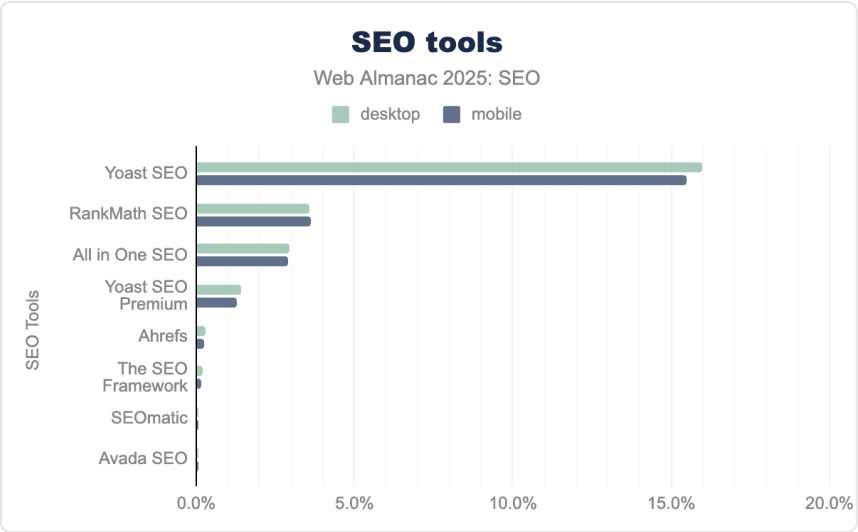


図5.9. 最も一般的なロボットディレクティブの実装

`meta` ロボットディレクティブのもう一つの要素は `name` 属性で、これらのディレクティブが特定のロボット/クローラーのユーザーエージェントを対象にしているかどうかを指定できます。たとえば、一部のクローラーが特定のルールを必要とし、他のクローラーがそうでない場合に動作をカスタマイズできます。

この属性内で最もよく名前が挙がるクローラーは2024年と一致しており：`bingbot`、`msnbot`、`googlebot`、`google-news` がデフォルトの `robots` とともに最も頻繁に表示されます。

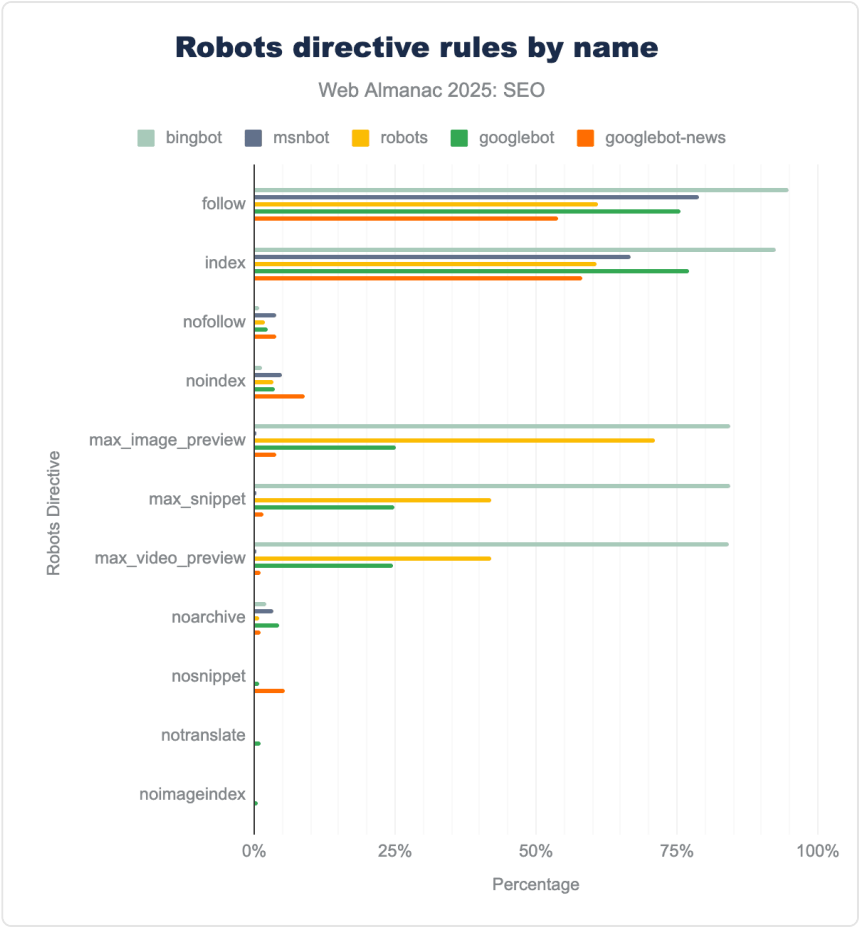


図5.10. 名前別ロボットディレクティブルール

`msnbot` は `bingbot` に置き換えられたレガシークローラーです。ロボットディレクティブへの継続的な存在は、古いクローラー名の更新や削除の遅れを示唆しています。この遅れの追加の証拠は、`max_image_preview`、`max_snippet`、`max_video_preview` などの新しいロボットルールが `googlebot` と `bingbot` には一般的に適用されているが、`msnbot` には適用されていないという事実に見られます。

`indexifembedded` タグ

現在よく確立されたMetaロボットルールである `indexifembedded` は、iframeコンテンツが埋め込まれているページの一部として扱われるようにしたい場合を指定できる非常に特定

のタグです。これを機能させるには `noindex` とペアにする必要があり、現在Googleのみがサポートするルールです。

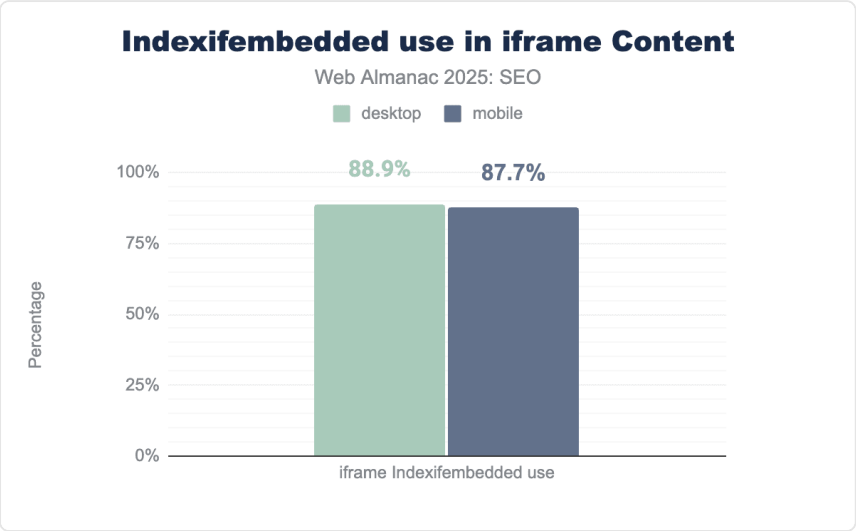


図5.11. `iframe` コンテンツでの `indexifembedded` の使用

iframeコンテンツ内では、`indexifembedded` ルールはほぼ90%の時間（デスクトップ88.9%、モバイル87.7%）で見つかります。興味深いのは、2022年から2024年にかけて使用率が上昇した後、2025年には99.9%の使用率から低下したことです。

無効な `<head>` 要素

検索エンジンのクローラーはコンテンツを解析する際にHTML標準に従います。発生しうる問題の一つは、ページの `<head>` 内に無効なHTML要素が見つかる場合です。これにより `<head>` が暗黙的に早期終了したと見なされ、残りのすべての `<head>` 要素がページの `<body>` 内に含まれるようになります。

SEOへの悪影響が最も大きいのは、`title`、`canonical` タグ、hreflang、Metaロボットディレクティブなどの重要なメタデータが無効な `head` 要素の下に配置されている場合（`body` 内に含まれることでそれらが機能しなくなります）です。

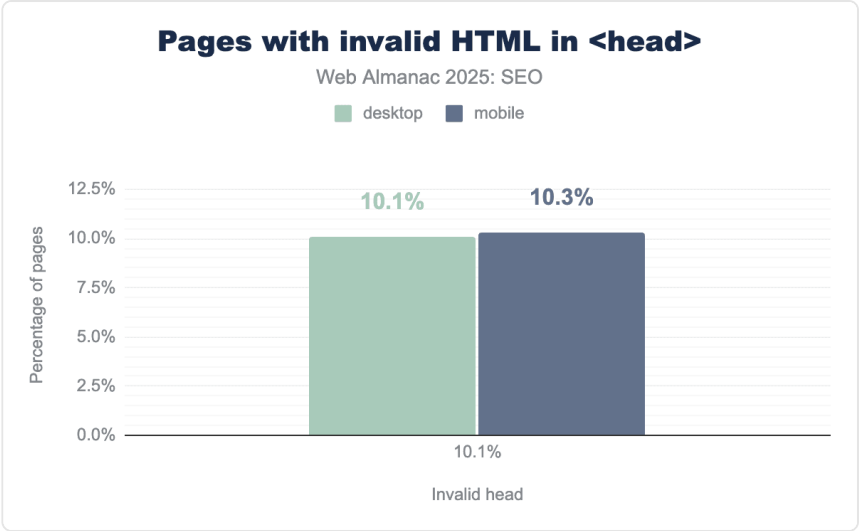


図5.12. <head> に無効なHTMLがあるページ

今年見つかった無効なhead要素は2024年のデータで見られた下降傾向を継続しており、前年比でページの <head> セクションの無効な要素が再び少なくなっています。

2025年には、デスクトップページで10.1%の無効な <head> 要素、モバイルページで10.3%の無効な要素が見られます。2024年のデータ（デスクトップ10.6%、モバイル10.9%）と比較すると、デスクトップで4.7%、モバイルで5.2%の低下を表しています。

「無効な」 <head> 要素の定義には、W3C標準に基づいていないページの <head> に含まれるものすべてが含まれます。Google検索のドキュメント[125]によると、 <head> で使用できる有効な要素は8つあります。これらは：

- <title>
- <meta>
- <link>
- <script>
- <style>
- <base>

125. https://developers.google.com/search/docs/crawling-indexing/valid-page-metadata#use-valid-elements-in-the-head-element

- `<noscript>`
- `<template>`

2025年に最も多く見られる無効な要素は2024年のデータと同じです：`<img>`、`<div>`、`<a>` タグ。

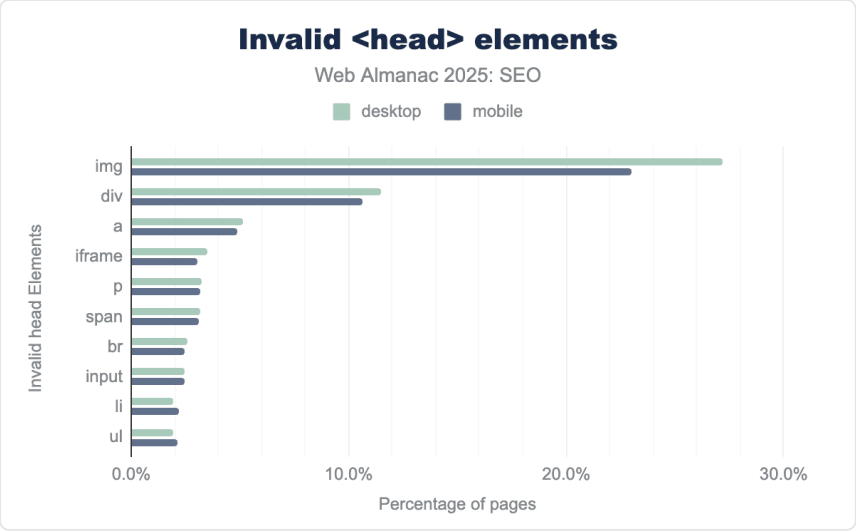


図5.13. 無効な `head` 要素

無効な `<img>` タグについては、2024年は2022年から急増が見られましたが、2025年の結果はより安定しています（今年はデスクトップ27.2%、モバイル23%に対し、昨年はデスクトップ29%、モバイル22%）。具体的には、デスクトップの使用率は低下し、モバイルは小幅に増加しています。

`<div>` タグはデスクトップ11.5%、モバイル10.6%で目立った変化はありません。`<a>` タグも同様に昨年からほとんど変化がなく、デスクトップ5.2%、モバイル4.9%です。

正規化

`canonical` タグは、重複またはほぼ重複するページが存在する場合に、どのバージョンが主要なページであるかを検索エンジンに伝えます。これらがなければ、クローラーはランキングシグナルを分散させ、クロールバジェットを無駄にし、検索結果に誤ったURLを表示する可能性があります。これらがあれば、すべての権限とインデックス優先度が単一の決定的なページに集中します。

`canonical` タグはディレクティブ（`meta` ロボットのような）ではなく、正しく実装することでランキング問題を最小化する支援シグナルを提供する「強いヒント」です。

`canonical` タグの使用がまた増加しており、昨年のモバイルとデスクトップの両方65%から、2025年にはデスクトップ68%、モバイル67%に増加しました。

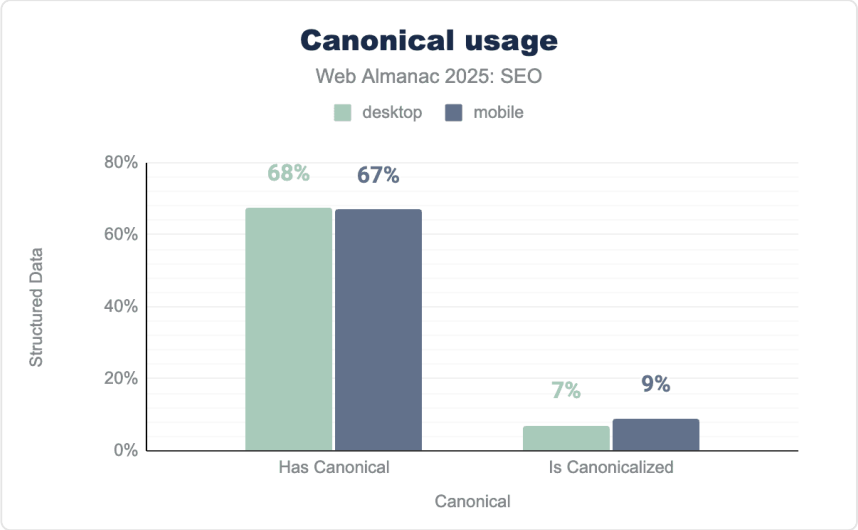


図5.14. `canonical` の使用状況

ページはそのcanonical URLが異なる場所を指す場合に「正規化されている」と言います。正規化されたページも今年増加しており、2024年のデスクトップ6%、モバイル8%から2025年にはデスクトップ7%、モバイル9%に増加しました。

`canonical` の実装

`Canonical` タグはMetaロボットと同様に、HTTPレスポンスの一部またはページの `head` 内に実装できます。各実装方法の背後にある動機は、プラットフォームや要件によって異なり、それぞれにメリットとデメリットがあります。

ページの `head` 内の `Canonical`（`rel="canonical"`）は最もよく使用されており（ほとんどのCMSで標準化されているため）、ページのcanonical URLを示します。

HTTP canonicalも同じことができ、PDFなどの他のファイルタイプにも使用できるという追加の利点があります。ただし、CMSユーザーには広く実装されておらず、より高度な技術的理解が必要です。

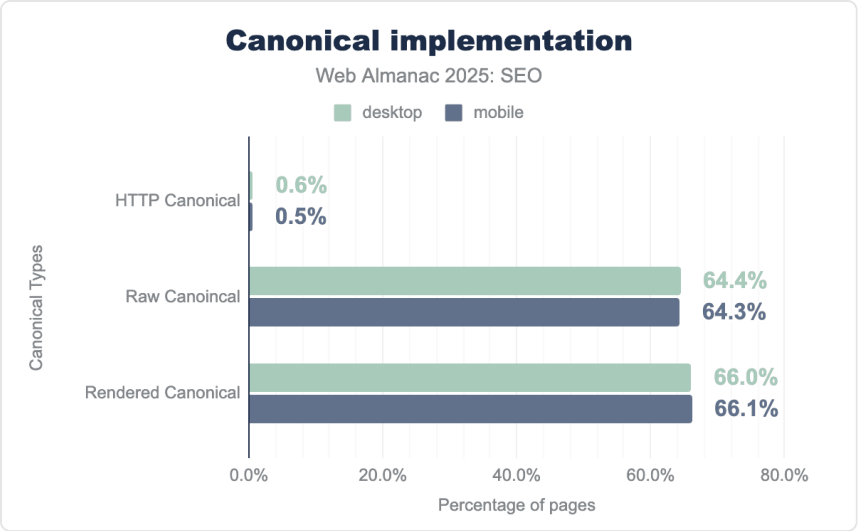


図5.15. `canonical` の実装

HTTPのcanonical実装は `rel="canonical"` タグと比較してまだ非常に低く、2025年にはデスクトップ0.6%、モバイル0.5%となっています。このデータは2024年の数値（デスクトップ0.7%、モバイル0.6%）からわずかな低下を表しています。

生のcanonicalタグとレンダリングされたcanonicalタグ

ページの `head` 内に配置される `rel=canonical` タグは、サイトの構築方法に応じて、生のHTML、レンダリングされたHTML、またはその両方に表示されます。ただし、最も信頼性の高いアプローチは、レンダリング後も変更されないことを保証するために、生のHTMLに `canonical` タグを含めることです。

JavaScriptで駆動されるページの場合、canonicalタグはページのレンダリングが完了した後にブラウザで挿入または更新されることがよくあります。生/レンダリングのcanonicalsの結果から、生のHTMLにcanonicalが欠如しているがレンダリングされたDOMに存在するケースは約2%のみです。これは、ほとんどのサイトが生のHTMLにcanonicalを含めることを選択し、ページレンダリング時に削除されないことを示しています。

2025年のデータでは、生のHTMLのcanonicalsの数値はデスクトップ64.4%、モバイル64.27%と2024年のデータと同等です。しかし、レンダリングされたcanonicalは2025年にデスクトップ66%、モバイル66.1%に増加しており、昨年のデスクトップ64%、モバイル65%と比較しています。

canonicalの矛盾

JavaScriptを使用して `head` 内でcanonicalタグをレンダリングすることは機能しますが、エラーの余地が増えるため、静的なサーバーサイドのcanonicalタグが一般的により安全なオプションです。canonicalタグが一致しない場合、その使用が損なわれ、予期しない結果を引き起こす可能性があります。canonicalの不一致は2024年の数値から0.1%低下し、デスクトップとモバイルの両方で0.8%から今年0.7%に減少しました。

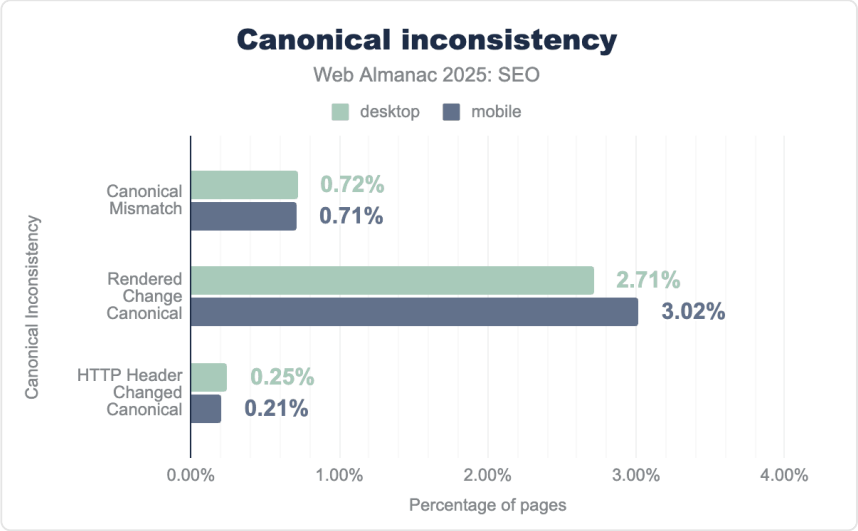


図5.16. `canonical` の不整合

ページエクスペリエンス

ページエクスペリエンスは、コンテンツの内容やランキングの高さを超えて、ユーザーがウェブページを操作する際に実際にどのように感じるかを捉えます。2020年にGoogleのランキングシステムの一部として導入され、これらのシグナルはページ速度、モバイルのレスポンシブ性、視覚的安定性、セキュリティなどの使いやすさの要因を測定します。Googleはその後、独立したページエクスペリエンスシステムをコアランキングフレームワーク[126]に統合しましたが、基礎となる指標はSEOとUXの両方のベストプラクティスを引き続き導いています。

2025年、データは技術的に成熟しているが、エクスペリエンスの品質においてまだ不均一なウェブを反映しています。HTTPSの採用はほぼ普遍的になり、モバイルフレンドリーさは

126. *https://developers.google.com/search/blog/2020/11/timing-for-page-experience*

差別化要因というよりも期待事項となり、Core Web Vitalsは実世界のパフォーマンスを定量化するための業界標準となっています。それでも、これらの進歩にもかかわらず、レンダリングの遅さからインタラクティビティの不一致まで、摩擦ポイントは依然として存在し、ユーザーエクスペリエンスはコンプライアンスと同様に一貫性についてであることをサイト所有者に思い出させています。

HTTPS

HTTPSはユーザーとウェブサイト間のデータ交換を保護するために導入され、情報を傍受や改ざんから守ります。後にGoogleのアルゴリズムで正式なランキングシグナル[127]となりました。

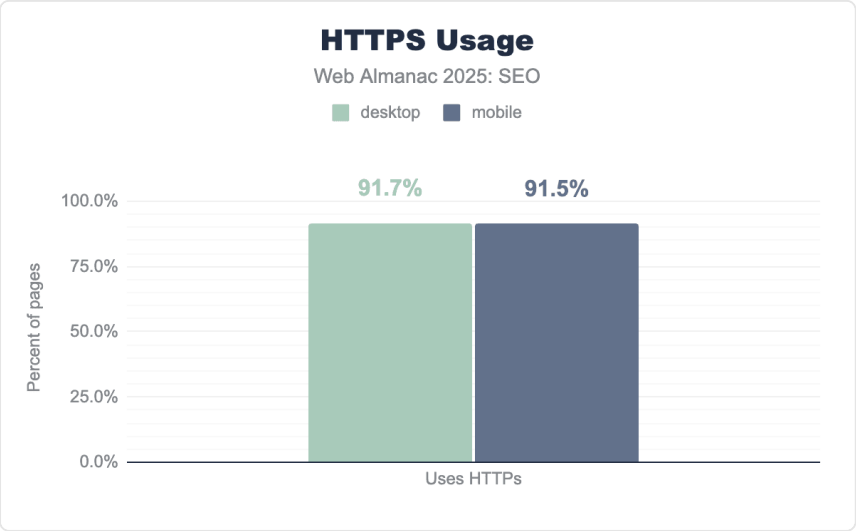


図5.17. HTTPSの使用状況

2025年、HTTPSの使用率はデスクトップページの91.7%、モバイルページの91.5%に達しており、2024年の約89%からわずかに増加しています。HTTPSがウェブのほとんどのデフォルトとなったため、成長は鈍化しています。残る小さなギャップは、まだ移行していない古いまたはメンテナンスが少ないサイトを表しています。その結果、HTTPSはユーザーと検索エンジンの両方にとって、差別化要因というよりもベースラインの期待として機能するようになっています。

それでも、完全な普遍的な暗号化は簡単ではありません。Googleの透明性レポート[128]は継続

127. *https://developers.google.com/search/blog/2014/08/https-as-ranking-signal*
128. *https://transparencyreport.google.com/https/overview*

的な障害を記録しています。一部の国や組織はHTTPSトラフィックをブロックまたは低下させ、技術的な能力が限られている組織は優先度を下げる場合があります。Googleの製品でさえ課題に直面しており、ユーザー所有ドメインの証明書管理（例：Blogger）は、それらのドメインがネイティブにHTTPSをサポートしていない場合に複雑になる可能性があります。これらの制約は、なぜ少数のサイトがまだ遅れをとっているかを説明するのに役立ちます。

モバイルフレンドリー

Googleは2023年にモバイルファーストインデックスへの移行を完了しました。つまり、ウェブサイトのモバイルバージョンが検索ランキングを決定するようになりました。以下のデータは、これに応じてサイトがモバイルフレンドリーなデザインプラクティスをどれだけ採用しているかを調べています。

`viewport` メタタグ

`viewport` メタタグは、ページがさまざまな画面サイズにわたってどのようにスケールおよび表示されるかを定義し、レスポンシブウェブデザインの重要な要素です。モバイルブラウジングの台頭とともに導入され、ユーザーが水平にズームやスクロールをすることなく、コンテンツがさまざまなデバイスにスムーズに適応することを保証します。

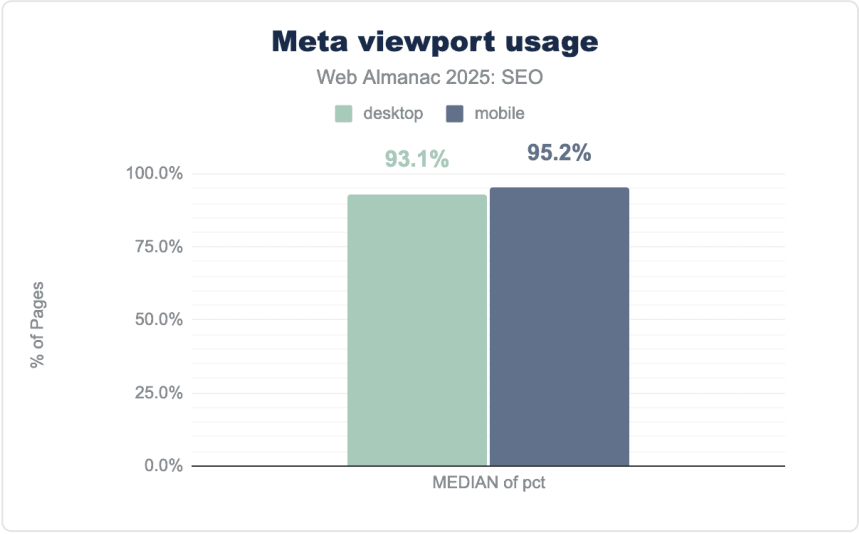


図5.18. `meta viewport` の使用状況

`viewport` メタタグの使用はほぼ普遍的で、デスクトップページの93.1%、モバイルページ

の95.2%に表示されています。これは、レスポンシブデザインの原則が現代のウェブサイトで広く使用されていることを示しています。デスクトップとモバイルの差が小さいことは、開発者が別々のモバイルとデスクトップのコードベースを維持するのではなく、すべてのデバイスで一貫したレスポンシブデザインアプローチを使用していることを意味し、モバイルフレンドリーなデザインプラクティスの成熟を示しています。

`Vary` ヘッダーの使用状況

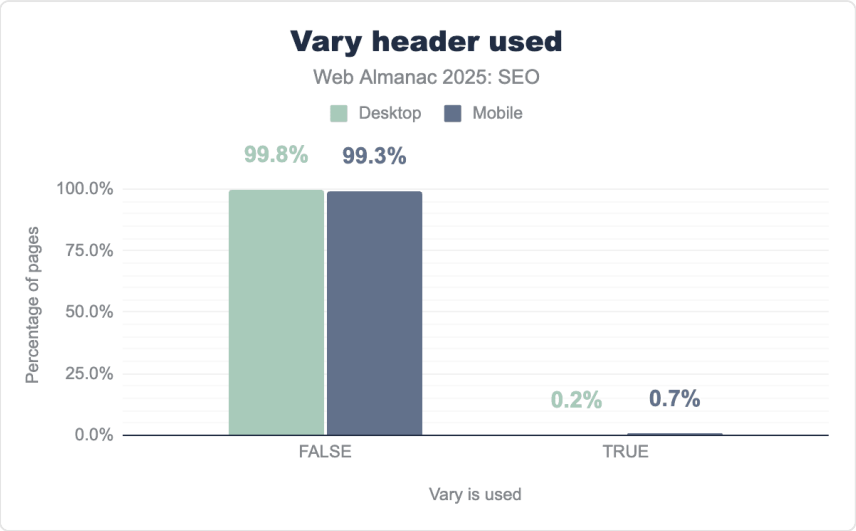


図5.19. `Vary` ヘッダーの使用

かつて動的配信の標準ツールだった `Vary: User-Agent header` は今やほぼ絶滅状態です。2025年には、ページの約1%しか含んでいませんが、ほぼすべてのサイトが統一されたレンダリングを選択しています。この変化は、`viewport` メタタグの普及（95%超）とGoogleのモバイルファーストインデックスへの移行と一致しており、サイトの別々のモバイルとデスクトップバージョンを維持する必要性を減少させました。ウェブは主にレスポンシブなシングルバージョンデザインに集約されています。

読みやすいフォントサイズ

読みやすさはモバイルエクスペリエンスの最も基本的な側面の一つです。ユーザーはズームや目を細めることなく快適にコンテンツを読めるべきで、アクセシビリティ標準もそれを強化しています。Web Content Accessibility Guidelines（WCAG[129]）によると、ボディテキスト

129. *https://www.w3.org/WAI/WCAG21/Understanding/resize-text.html#*

は少なくとも16px（1rem）で、レイアウトや機能を損なわずに200%までスケールできる必要があります。

Lighthouseの読みやすいフォントサイズ監査[130]は同様のベンチマークを使用し、可視テキストの60%未満が `12px` を超えるページにフラグを立てます。2025年には、モバイルページの92%がこのテストを通過しており、2024年のパフォーマンスを反映し、ほとんどのサイトがホームページとインナーページの両方で基本的な読みやすさ標準を満たしていることを示しています。

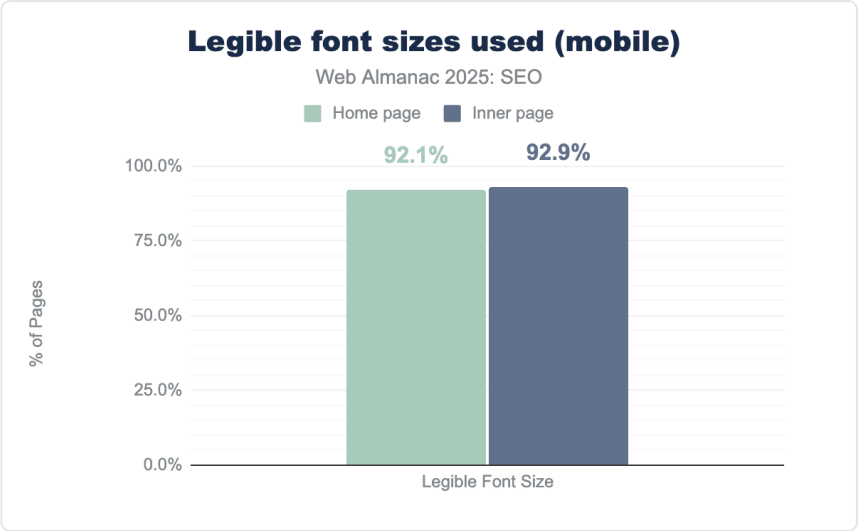


図5.20. 使用されている読みやすいフォントサイズ（モバイル）

Core Web Vitals

Core Web Vitalsは、読み込み（Largest Contentful Paint、LCP）、インタラクティビティ（現在はInteraction to Next Paint、INPで測定）、視覚的安定性（Cumulative Layout Shift、CLS）に焦点を当て、実際のユーザーがページをどのように体験するかを測定します。

130. *https://developer.chrome.com/docs/lighthouse/seo/font-size*

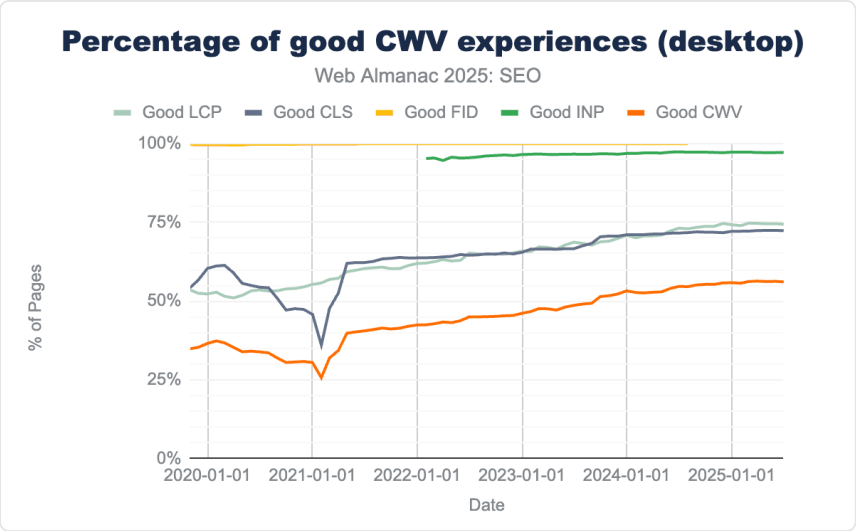


図5.21. 良好なCore Web Vitalsエクスペリエンスの割合（デスクトップ）

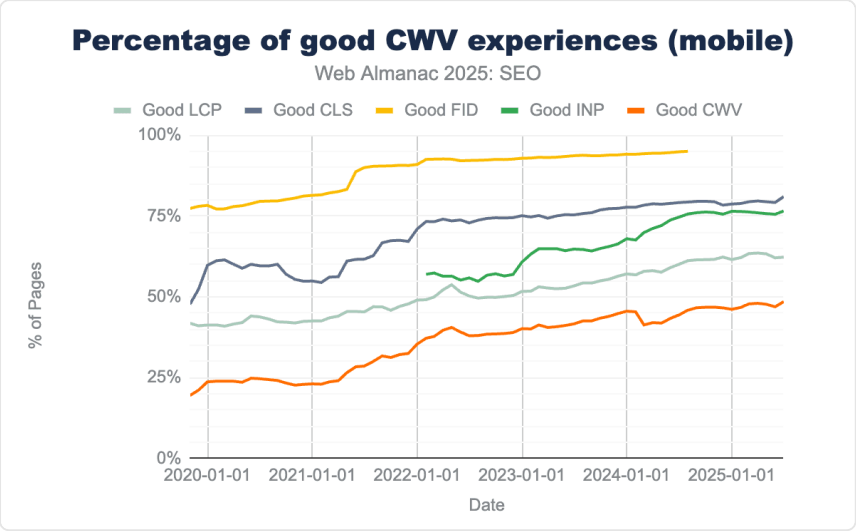


図5.22. 良好なCore Web Vitalsエクスペリエンスの割合（モバイル）

2025年、Core Web Vitals全体のパフォーマンスは数年間の着実な前年比改善を経てほぼ安定しています。デスクトップはCore Web Vitalsパフォーマンスで引き続きリードし、約60%のページが「良好」な総合Core Web Vitalsエクスペリエンスを提供しており、モバイルの50%強と比較しています。この差は、より高速なハードウェア、より強力な接続、レイアウト制約が少ないといったデスクトップの固有の優位性を反映しています。

両プラットフォームにわたって、CLSが最も強い指標で、約75%のページが合格しています。しかし、LCPはモバイルで55〜60%近くにとどまり続けており、読み込み速度が今日のウェブサイトにとって最も持続的なCore Web Vitalsの課題であることを示しています。研究が示すように[131]、これはユーザー満足度に影響し、特にAI駆動の環境でオンラインコンバージョンにも影響を与える可能性があります。

2024年3月、特にモバイルで見られる目立つ低下は、Googleの2024年3月のFirst Input Delay（FID）からInteraction to Next Paint（INP）への置き換え[132]への移行を反映しています。INPは展開前にパフォーマンスツールに表示され始め、FIDがしばしば見落としていたレスポンシブネスの問題を露呈する、より厳しいしきい値を導入しました。この段階的な変化は「良好な」ユーザーエクスペリエンスと見なされるバーを引き上げ、変更が発効したときの変動を説明しています。

画像の `loading` プロパティの使用状況

`loading` 属性はブラウザに画像をいつ取得してレンダリングするかを伝えます。速度とリソース使用のバランスをとるのに役立つ組み込みのパフォーマンスヒントです。動作を決定する2つの主な値があります：

- `lazy`: 画像が画面に表示されるまで読み込みを遅延させ、初期ページの重さを減らし、体感パフォーマンスを向上させます。
- `eager`: ページが読み込まれるとすぐに画像を取得し、ヒーローバナーやロゴなどのアバブザフォールドビジュアルの即時表示を確保します。

この属性により、開発者は異なるコンテキストで読み込み動作を微調整し、重要なビジュアルを優先しながら重要度の低いものを後回しにして読み込み時間と帯域幅効率を向上させることができます。

131. *https://www.speedcurve.com/blog/psychology-site-speed/*
132. *https://searchengineland.com/google-replace-fid-inp-core-web-vitals-414546#:~:text=On%20the%20left%2C%20long%20tasks,rankings%2C%E2%80%9D%20according%20to%20Splitt.*

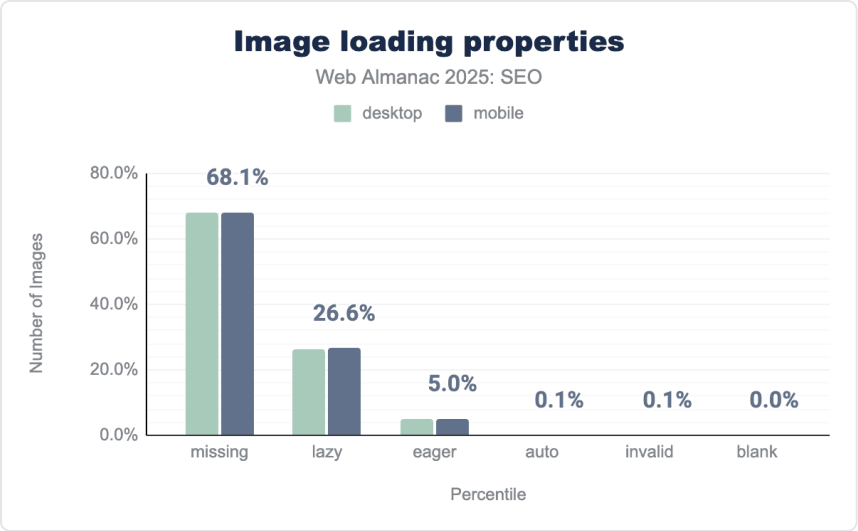


図5.23. 画像の `loading` プロパティ

2025年、約68%の画像（デスクトップとモバイルの両方）に画像読み込み属性が設定されておらず、読み込み動作はブラウザのデフォルトに任されています。一方、約26%が遅延読み込みを使用し、わずか4〜5%のみが明示的にeager読み込みを使用しています。

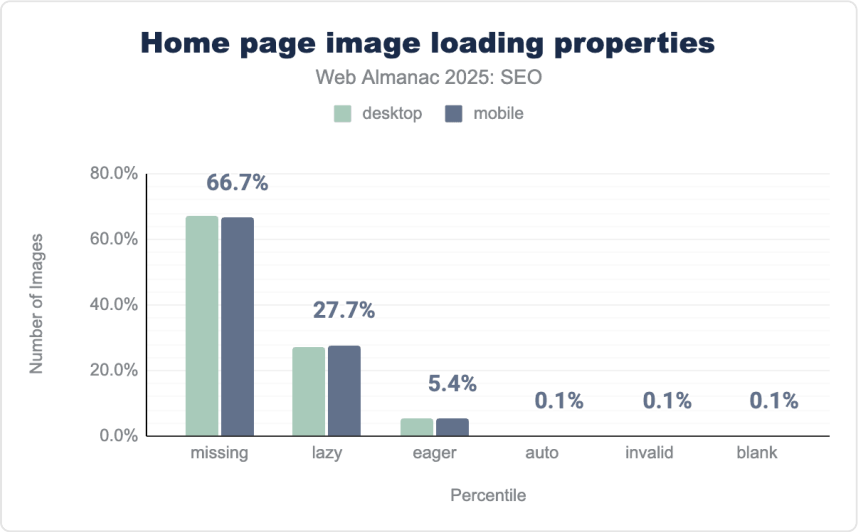


図5.24. ホームページの画像 `loading` プロパティ

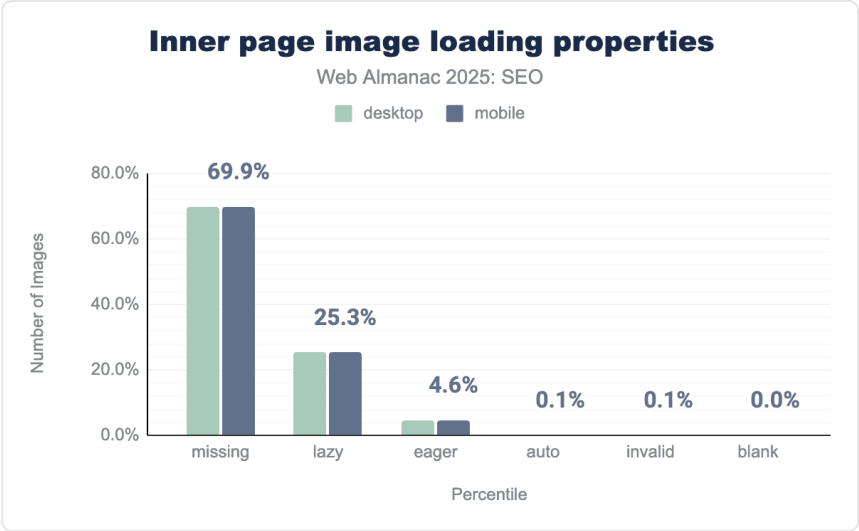


図5.25. インナーページの画像 `loading` プロパティ

このパターンはページタイプ全体で一貫しています：

- ホームページ: デスクトップ27.3%、モバイル27.7%が遅延読み込み
- インナーページ: デスクトップ25.6%、モバイル25.3%が遅延読み込み

画像読み込み属性の欠如が多いことは、遅延読み込みの採用がパフォーマンス重視のサイトに集中している一方、多くの他のサイト（おそらくレガシーまたはCMS駆動のサイト）が明示的な読み込み戦略を実装していないことを示唆しています。`loading="lazy"` のほぼ普遍的なブラウザサポートとレスポンシブデザインを考慮すると、大多数の画像が体感パフォーマンスを向上させ帯域幅使用を削減するために明示的な遅延読み込みの恩恵を受けられます。

iframeの `lazy` 読み込みと `eager` 読み込み

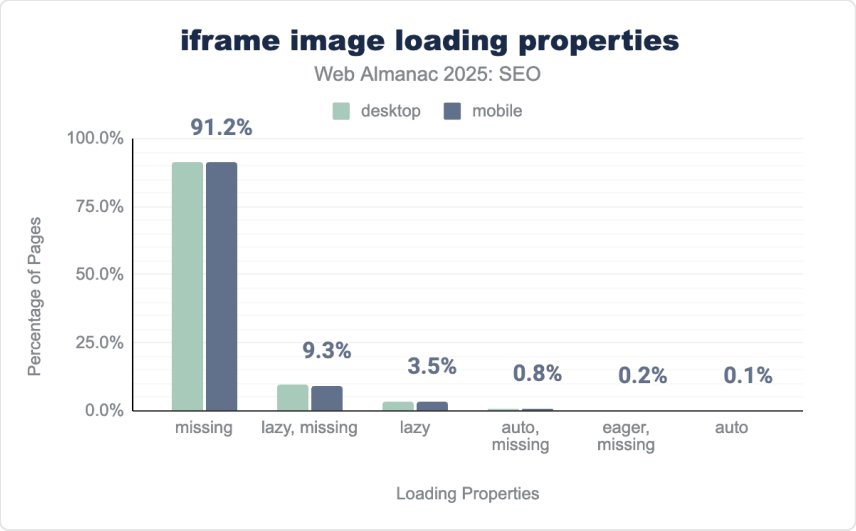


図5.26. `iframe` の画像読み込みプロパティ

iframeに `loading="lazy"` を追加するだけで、ビューポートの近くまで読み込みが遅延され、YouTube埋め込み[133]で約500KB、Facebookウィジェット[134]で約200KBを節約できます。しかし、2025年にiframeの読み込み属性は依然として著しく十分に活用されておらず、デスクトップページの91.5%、モバイルページの91.2%がiframe読み込みプロパティを宣言していません。これは2024年のデスクトップ92.8%、モバイル92.6%からわずかに低下しています。

明示的なiframe遅延読み込み属性は約13%のページに表示されており（3.5%が単独、9.3%が欠如宣言と混在）、しばしば宣言のないiframeと一緒に一貫性なく適用されています。前年と同様に、このパターンは出版社が制御を制限されている広告、動画、分析ダッシュボードなどの埋め込みサードパーティ要素の優位性を反映していると思われます。明示的な `eager` 宣言は事実上存在しないままで（デスクトップとモバイルの両方で0.2%）、単にブラウザのデフォルトを再述するだけです。大幅なパフォーマンス向上の機会があるにもかかわらず、iframeの遅延読み込みの採用は画像の遅延読み込みよりもはるかに遅れています。

この不一致は主に、広告、動画プレーヤー、分析ダッシュボードなどのサードパーティの埋め込みがiframeの使用を支配している方法を反映しています。これらの要素は外部で制御されているため、出版社は読み込み動作を定義したり属性を直接追加したりする能力が限られています。多くの場合、サードパーティのスクリプトは読み込み後に動的にiframeを注入す

133. *https://web.dev/articles/iframe-lazy-loading*
134. *https://web.dev/learn/performance/lazy-load-images-and-iframe-elements*

るため、クローラーと監査はiframeが存在していても `loading` 属性を検出しません。その結果、表面上「欠如している」属性の一部は、真の省略ではなく測定における可視性のギャップを表している可能性があります。

これはまた開発者の優先度付けとも関係しています。チームは通常、制御が難しく増分的な利益が小さいiframe動作に対処する前に、ファーストパーティアセット（画像、スクリプト、Core Web Vitals）のパフォーマンスの最適化に焦点を当てます。

オンページ

ページ構造は現在のAI駆動の環境においても依然として重要です。タイトル、メタディスクリプション、見出しは意味と意図を伝え、メディア属性は明確さとアクセシビリティを強化します。2025年のデータは、これらの要素の使用全体における安定性を示しており、大きな変化ではなく漸進的な調整を反映したわずかなシフトのみです。

メタデータ

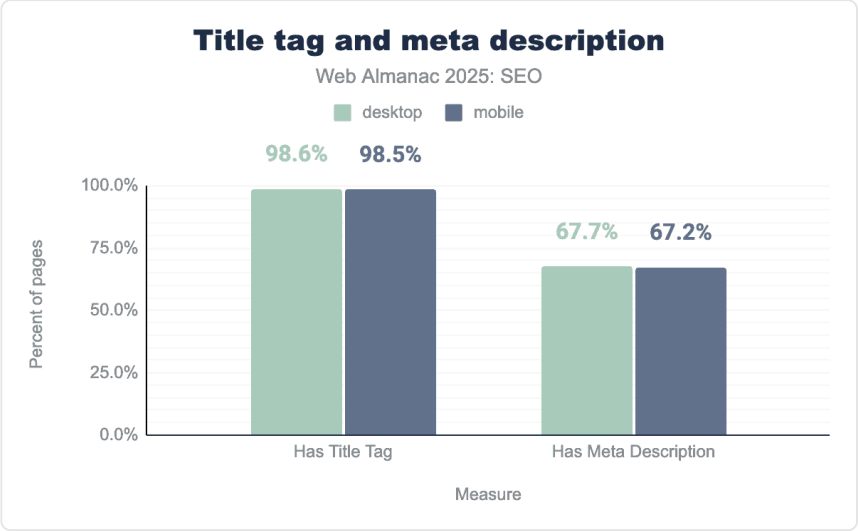


図5.27. `title` タグとメタディスクリプション

`title` タグの採用はほぼ普遍的なままで、2025年にはデスクトップページの98.62%、モバイルページの98.54%にわずかに増加しており、2024年のデスクトップ98.0%、モバイル98.2%から上昇しています。この安定性は、検索結果でタイトルが広く書き換えられているにもかかわらず、SEO戦略における要素の永続的な役割を反映しています。出版社はまだ、

表示制御に関係なくtitleタグを基本的なものとして見ているようです。

一方、メタディスクリプションは異なる軌跡をたどっています。2022年のページの71%への表示から2024年のデスクトップ66.7%、モバイル66.4%へと低下した後、2025年にはデスクトップ67.7%、モバイル67.2%にわずかに回復しました。採用率は2022年の以前の水準には完全に回復していませんが、今年の上昇はGoogleが書き換えることがあっても、出版社がクリックスルーの最適化とクロスプラットフォームの一貫性のためにメタディスクリプションに価値を見出し続けていることを示唆しています。

`title` 要素の長さ

`title` タグはユーザーとアルゴリズムの両方にとってページのトピックの最も明確なシグナルの一つですが、その効果は部分的に長さに依存します。短すぎるとコンテキストが不足し、長すぎると検索結果で切り捨てられるリスクがあります。2025年のデータは、出版社が2024年からほとんど変化なく、安定した中間点の周りにクラスタリングし続けていることを示しています。

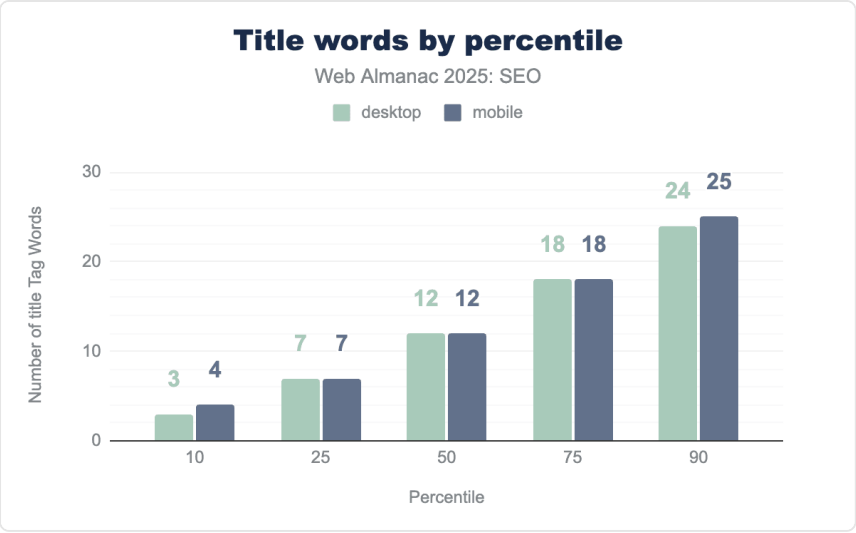


図5.28. パーセンタイル別のtitleの単語数

中央値のtitleタグはデスクトップとモバイルの両方で12単語を含み、昨年から変わっていません。75パーセンタイルでは18単語に増加し、90パーセンタイルではデスクトップで24単語、モバイルで25単語に延びます。

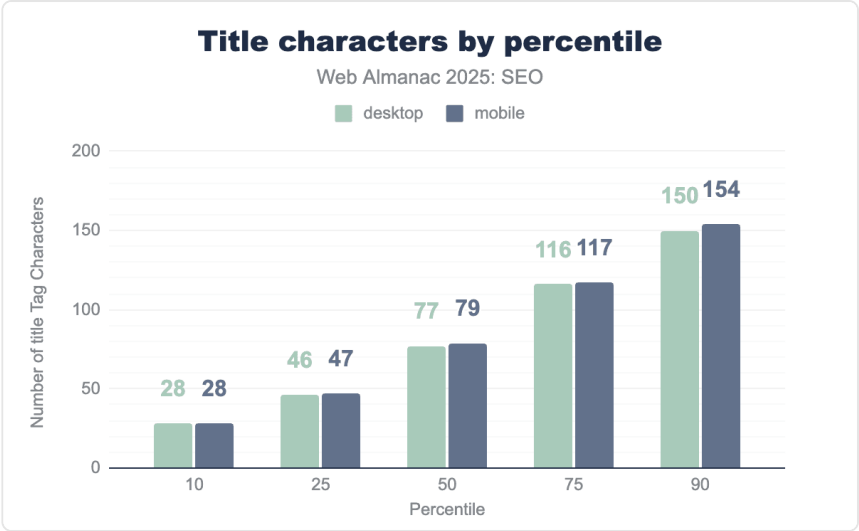


図5.29. パーセンタイル別のtitleの文字数

中央値のtitleの長さはデスクトップで77文字、モバイルで79文字で、2024年とほぼ同じです。75パーセンタイルでは116〜117文字に上昇し、90パーセンタイルでは150〜154文字に達します。

これらの数値は、よく引用されるtitleの「安全な範囲」である50〜60文字を大幅に上回っており、表示が切り捨てられても出版社がキーワードカバレッジとコンテキストを優先していることを示唆しています。

2024年から2025年にかけてこのパターンが継続していることは、Googleの書き換えプラクティスを反映している可能性もあります。タイトルはSERPで頻繁に変更されるため、出版社はピクセルパーフェクトな表示制限に合わせて細かく調整する動機が少なくなっています。

メタディスクリプションの長さ

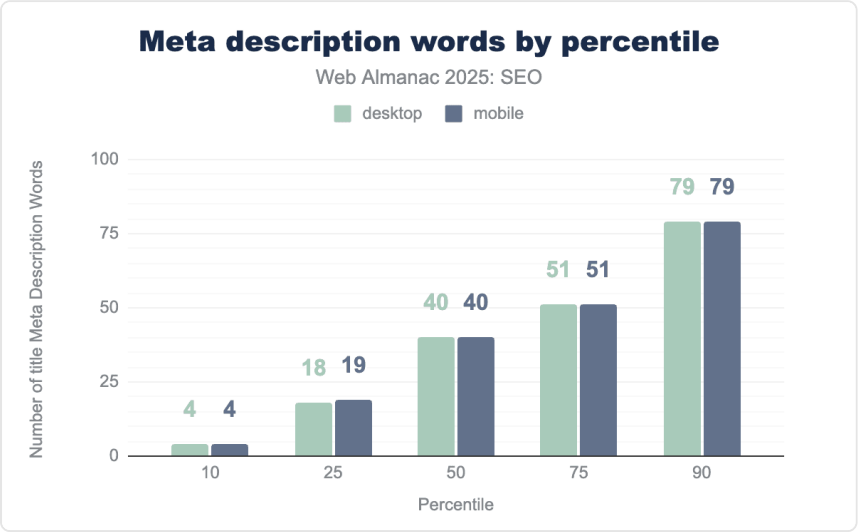


図5.30. パーセンタイル別のメタディスクリプションの単語数

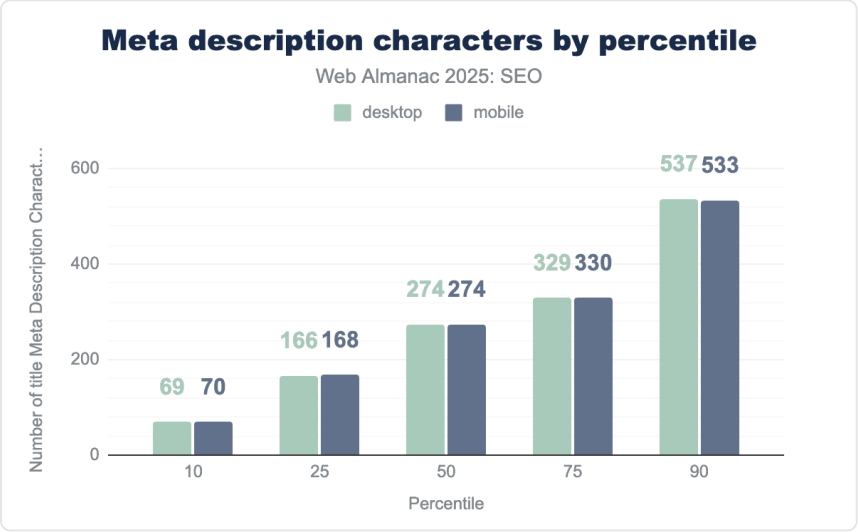


図5.31. パーセンタイル別のメタディスクリプションの文字数

分布パターンは2022年から2024年の急激な成長の後、ほぼ安定しています。今年のデータは以下を示しています：

- **中央値（50パーセンタイル）:** デスクトップとモバイルの両方で40単語、274文字と、2024年から変わりませんが、2022年の中央値19単語、約135文字の2倍以上です。
- **10パーセンタイル:** 4単語、約69〜70文字と、昨年と同じです。
- **25パーセンタイル:** 18〜19単語、166〜168文字。
- **75パーセンタイル:** 51単語、329〜330文字。
- **90パーセンタイル:** 79単語、533〜537文字。

この分布は、出版社が快適な範囲に落ち着いていることを確認しています：内容を提供するのに十分な長さで、一般的には約80単語または約540文字以下に抑えられています。Google が頻繁にスニペットを書き換えても、適切に構造化されたディスクリプションを維持することで、サイト所有者はコンテンツが他の場所でどのように表示されるかに一定の影響力を持てます。

実際には、スニペットの表示はピクセル幅によって制限されており（デスクトップで約920〜980px[135]、モバイルで680px）、切り捨ては文字幅によって異なります。

このパターンが続けば、適切に構造化されたメタディスクリプションは従来のSERPシグナル以上のものになります：ページがAI駆動の要約や引用のために表示されるかどうかを決定するのに役立ちます。

見出しタグ

適切に構造化された見出しはテキストをフォーマットするだけでなく、ページの論理的な流れをマッピングし、読者とクローラーの両方を主なアイデアに案内します。2025年のデータは、すべての主要タグにわたる安定性を示しており、昨年からわずかなシフトのみです。

135. *https://ux.stackexchange.com/questions/125770/does-the-980px-screen-width-rule-still-stand*

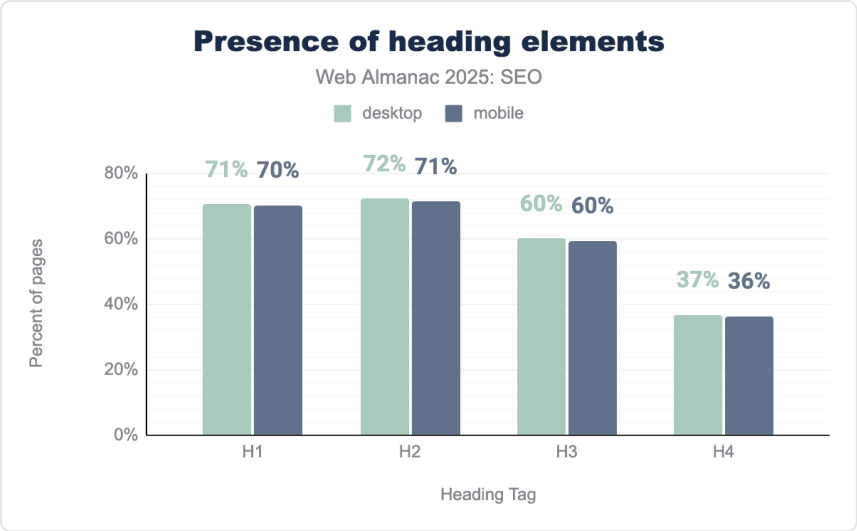


図5.32. 見出し要素の存在

- `<h1>`: デスクトップページの71%、モバイルページの70%に存在します。空でない使用はわずかに低く、両方で66%です。
- `<h2>`: 最も広く採用されており、デスクトップページの72%、モバイルの71%で、ほぼすべてが空でありません。
- `<h3>`: 60%のページで使用されており、58%が空でなく、2024年から変わっていません。
- `<h4>`: デスクトップページの37%、モバイルの36%に存在し、35〜36%が意味のあるコンテンツを含んでいます。

見出しタグの採用は明らかに頭打ちになっています。以前の年には `<h1>` の成長と `<h3>` / `<h4>` の低下が見られましたが、2025年は安定したパターンを確認しています。

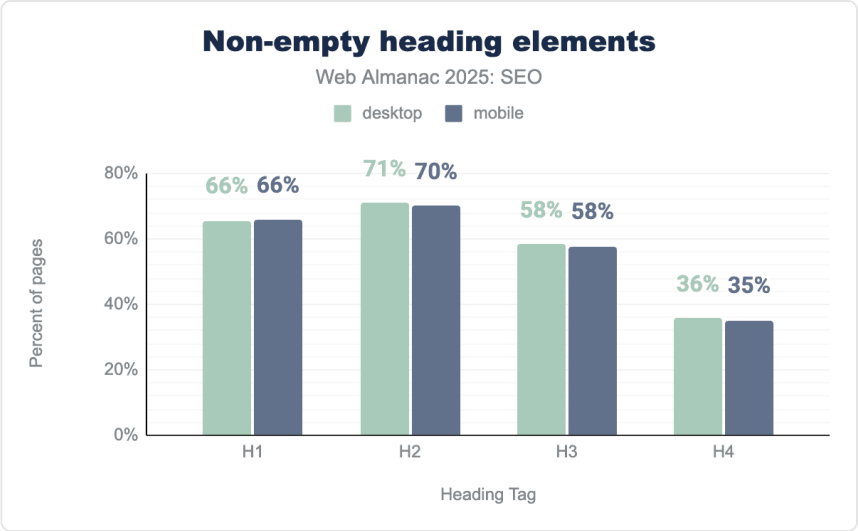


図5.33. 空でない見出し要素

空でないデータは特に示唆に富んでいます。「存在する」と「空でない」のギャップは今やわずか数パーセントポイントで、2022年（デスクトップの `<h1>` の約6%が空）の大きなギャップと比較しています。これは、空の見出しが例外ではなく標準となった、よりクリーンでより意味的に構造化されたマークアップを示唆しています。

アクセシビリティとSEOを超えて、この見出し構造の改良はより広いトレンドとも一致しています：出版社がAI駆動プラットフォームのためにオンページの明確さを最適化しています。AIの概要、要約ツール、引用システムがクリーンなセマンティック階層に依存して情報を抽出・帰属させるようになるにつれ、適切に構造化された見出しはページのアイデアがAI出力で正確に表現・クレジットされることを保証する一部となっています。

それでも進歩には限界があります。`<h1>` はページの70%に表示されていますが、空でないのは66%のみです。この4%のギャップは上限効果を示しており、採用率が100%に達することは考えにくい自然な上限です。シングルページアプリやミニマリストのホームページなど、一部のデザインは意図的に意味のある `<h1>` をスキップします。これらのケースはエラーではなくデザインの選択を反映しています。

画像

2025年の画像使用データは以前の年との幅広い一貫性を示していますが、サイトの上位層に向けて画像使用が劇的にスケールアップすることも強調しています。

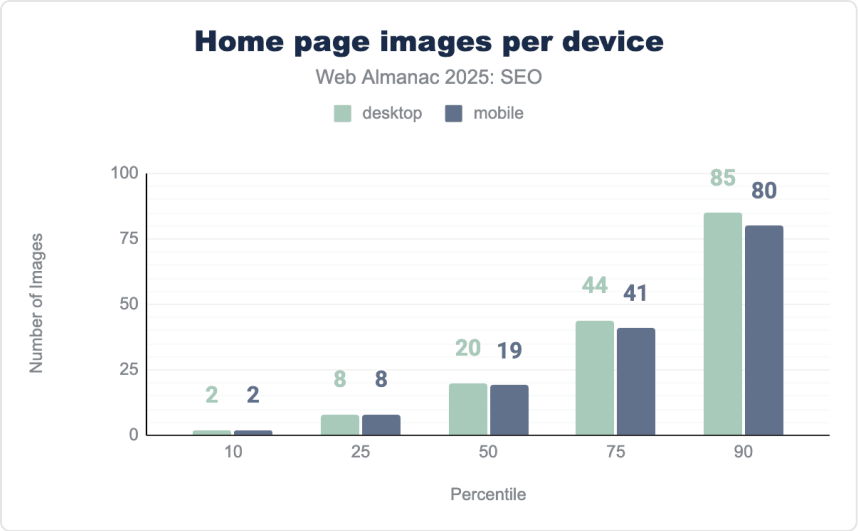


図5.34. デバイスごとのホームページの画像数

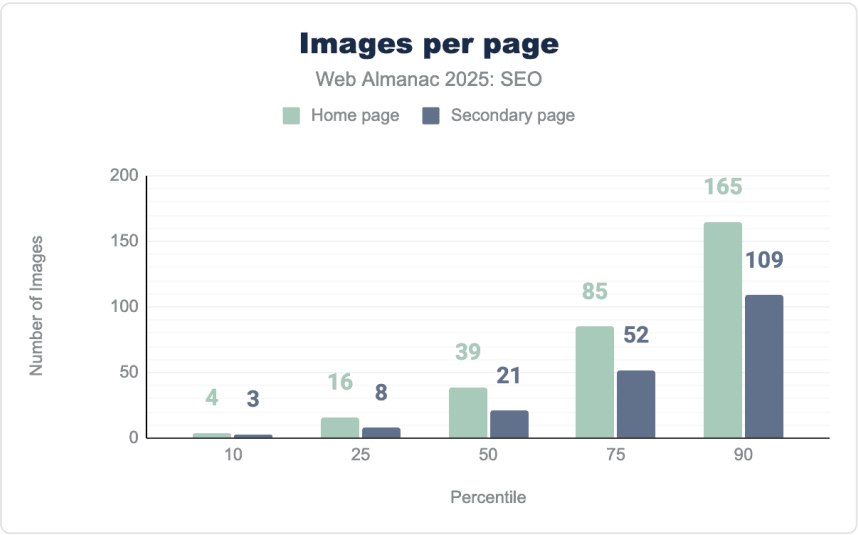


図5.35. ページごとの画像数

- 中央値（50パーセンタイル）では、ホームページにはデスクトップで20枚、モバイルで19枚の画像があります。
- 75パーセンタイルでは、画像数はデスクトップで44枚、モバイルで41枚に増加します。

- 90パーセンタイルでは、画像の使用はデスクトップで85枚、モバイルで80枚にさらに跳ね上がります。

比較すると、25パーセンタイルはわずか8枚の画像で、ホームページデザインの幅広いバリエーションを示しています。しかし分布全体にわたって、デスクトップとモバイルの数値は密接に一致しており、レスポンシブデザインプラクティスがデバイス間で画像使用を一貫させていることを示唆しています。

`alt` 属性

`alt` 属性はアクセシビリティの中心であり、検索での画像インデックスもサポートしています。今年は採用に着実な改善が見られ、全体的に欠如している属性が減っていますが、注目すべき空白値のわずかな増加もあります。

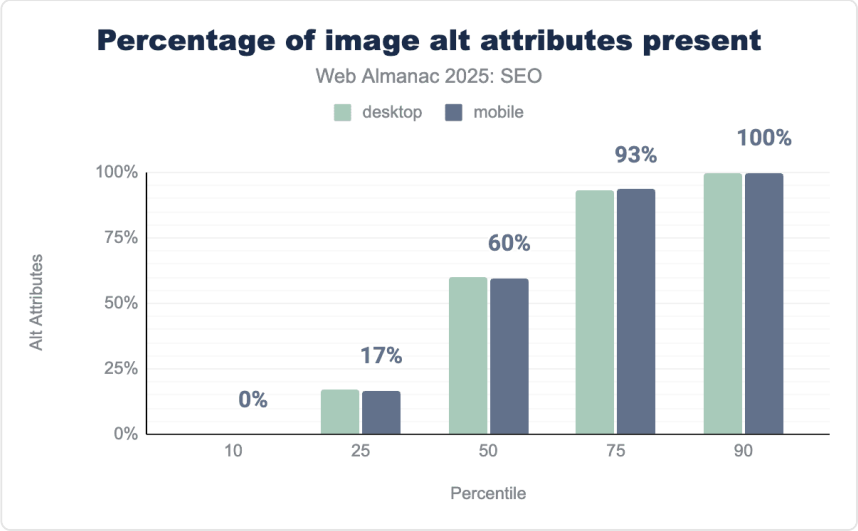


図5.36. 画像のalt属性が存在する割合

中央値では、モバイルページの画像の60%が `alt` 属性を含んでおり、2024年の58%から増加しています。90パーセンタイルでは採用率が100%に達し、出版社の相当数がすべての画像に一貫してalt テキストを適用していることを示しています。

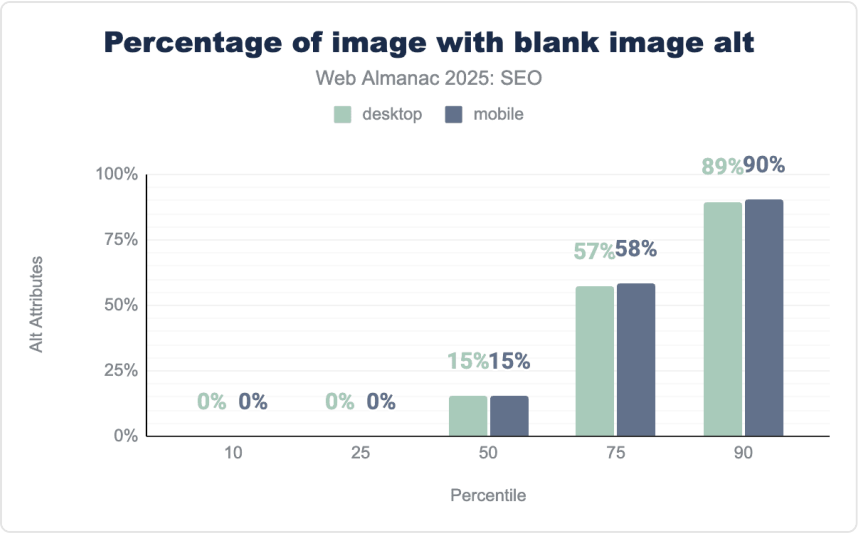


図5.37. 空白のimage altを持つ画像の割合

中央値では、モバイルページの画像の15%が空白の `alt` 値を持っており、2024年の14%からわずか1ポイント増加しています。75パーセンタイルでは58%が空白で、90パーセンタイルでは90%が空白となり、いずれも昨年より1ポイント高くなっています。増加は小さいですが、すべての採用が意味のある説明に転換されているわけではないという懸念を示しています。

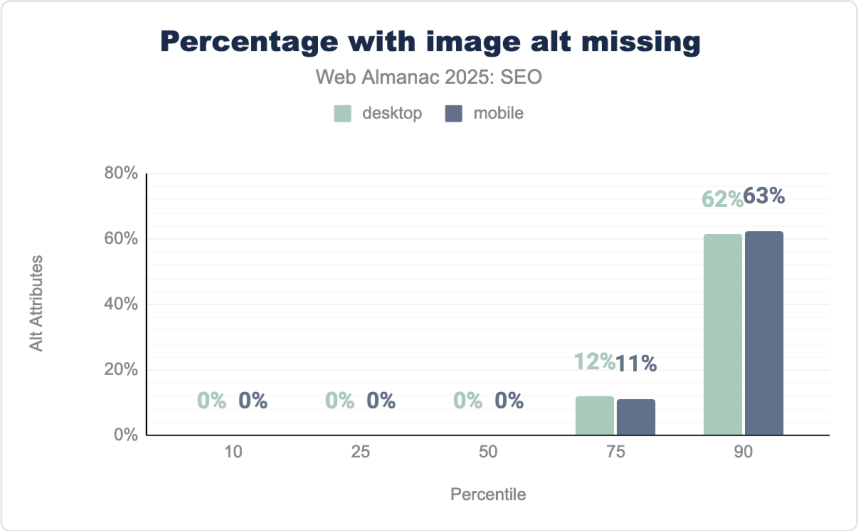


図5.38. altが欠如している画像の割合

中央値のページは再び `alt` の欠如が見られず、12〜13%のページが完全に欠如していた2022年からのポジティブなトレンドが続いています。

より広い変化は「欠如」から「空白」へのシフトのようです。画像には属性が配置される可能性が高くなっていますが、常に説明的なコンテンツが伴うわけではありません。空白は省略より望ましいですが、スクリーンリーダーと検索エンジンに対して限られた価値しか提供しません。

空白の増加は劇的ではなく小さく安定しているものの、技術的なコンプライアンスと意味のあるアクセシビリティの差を依然として浮き彫りにしています。2025年のホームページ画像数が最少8枚から80枚以上にわたることを考えると、思慮深いaltテキストの重要性は増すばかりです。ウェブは明らかに正しい方向に進んでいますが、課題はカバレッジの進歩がアクセシビリティと意味においても進歩をもたらすことを保証することです。

動画

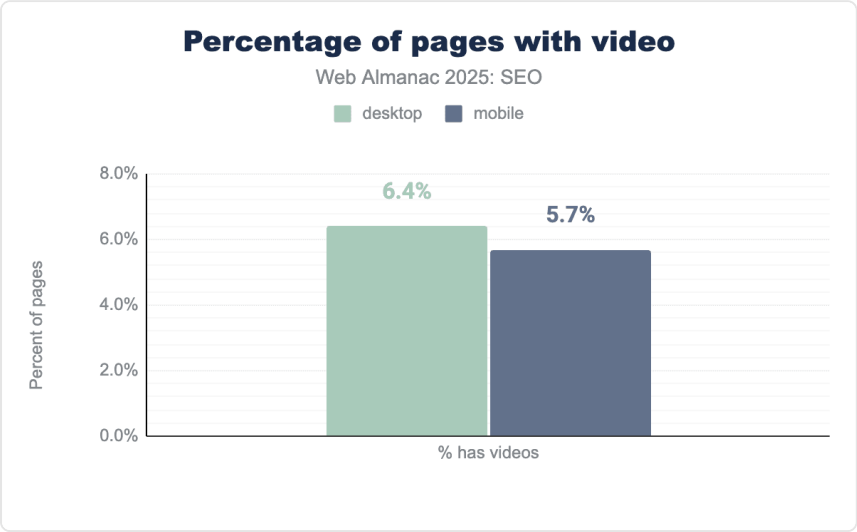


図5.39. 動画があるページの割合

ウェブ上での動画の採用は続いて成長していますが、緩やかです。2025年には、デスクトップページの6.40%、モバイルページの5.68%に動画要素が含まれており、2024年のデスクトップ5.87%、モバイル5.13%からわずかに増加しています。デスクトップページはまだモバイルよりも動画を多く掲載しており、昨年と同じギャップを繰り返しています。

この控えめな成長は、画像と比較して動画の制作および実装コストが高いことを反映しているかもしれません。しかし、長期的なトレジェクトリは上昇傾向のままです。SEOがより包括的になり、ブランディングがウェブ全体で最も強いシグナルの一つとして浮上するにつれ、動画は権威とクロスチャネルの視認性を強化する上でより大きな役割を果たす可能性があります。

より大きな課題は最適化です。動画がより頻繁に登場している一方で、2024年にVideoObjectマークアップを使用したのはページの0.9%のみで、ほとんどの動画コンテンツがリッチリザルトに見えないままになっています。構造化データの採用が増えるまで、動画のSEOポテンシャルの多くは未活用のままです。

リンク

リンクは依然として重要性の重要なシグナルです。しかし、リンクが多いだけでランキングが上がる時代はほぼ終わりました。今日のリンクシグナル（内部・外部）はコンテンツの品

質、関連性、ユーザーエクスペリエンスとバランスよく組み合わされています。

2025年のデータは、リンクプラクティスが成熟していることを示しており、説明的なアンカー、内部ナビゲーション、外部参照において安定したパターンが見られ、関係を修飾するためのrelなどの属性の選択的な使用もあります。

非説明的なリンク

2025年、ほとんどのサイトが「ここをクリック」や「もっと読む」などの漠然とした表現よりも説明的なアンカーテキストを提供し続けています。これはLighthouseのテスト合格率に反映されており、ホームページとインナーページの両方で高い水準を維持しています。

- 2024年: ホームページはデスクトップ84%、モバイル91%で合格し、インナーページはデスクトップ86%、モバイル92%とより強い結果でした。
- 2025年: ホームページはデスクトップ84.64%、モバイル86.64%とほぼ変わらず、しかしインナーページはデスクトップ92.42%、モバイル93.45%に改善しました。

ホームページとインナーページの差は持続しており、インナーページが一貫して約6〜9ポイント高くなっています。これはホームページが汎用のナビゲーションとプロモーション用CTAに依存しやすい一方、インナーページは自然とより説明的なコンテキスト・コンテンツ駆動のリンクを使用しているためと考えられます。デバイス間のパリティは近く、レスポンシブデザインプラクティスがデバイス間のアクセシビリティ標準を効果的に維持していることを示しています。

全体として、説明的なリンクプラクティスは成熟しており、前年比のわずかな変化のみです。残る機会は、漠然としたCTAがまだ最も一般的なホームページリンクの明確さを改善することにあります。

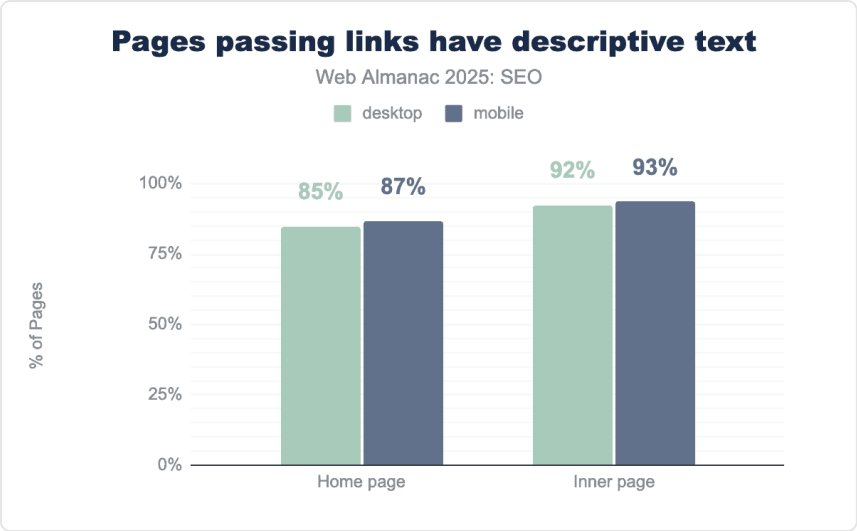


図5.40. 説明的なテキストを持つリンクが通過するページ

発信リンク

2025年の発信リンクは、内部および外部リンクの両方で安定したパターンを示しています。内部リンクが優勢で、中央値ではデスクトップページが同一サイトへの43リンク、モバイルページが39リンクを含んでいたのに対し、90パーセンタイルではそれらの数字が174と161に跳ね上がりました。

上位に向けた急激な上昇は、大規模または複雑なサイト（ニュースメディア、eコマースプラットフォーム、エンタープライズポータルなど）がメガメニューや密集したフッターを通じて広範なナビゲーションを表示する方法を反映しているかもしれません。デスクトップページは一貫してモバイルよりも数リンク多く表示していますが、開発者は使いやすさのために小さな画面でメニューを合理化しながらも、主要なパスウェイは維持しています。

外部リンクははるかに少ないままです。中央値のページはデスクトップとモバイルの両方でわずか6つの発信リンクを含んでおり、90パーセンタイルでもその数は25と24にしか達しません。これは2022年を反映した2024年でも記録されたフラットなトレンドを継続しています。

内部リンクは長年にわたって着実に成長している一方、外部リンクは保守的なままで、おそらく出版社がユーザーを自分のエコシステム内に留めることに焦点を当てていることを反映しています。合わせて、これらの数値は成熟した均衡を示唆しています。内部リンクはナビゲーションとクロール可能性をサポートするために成長する一方、外部リンクは控えめなレベルで安定しています。

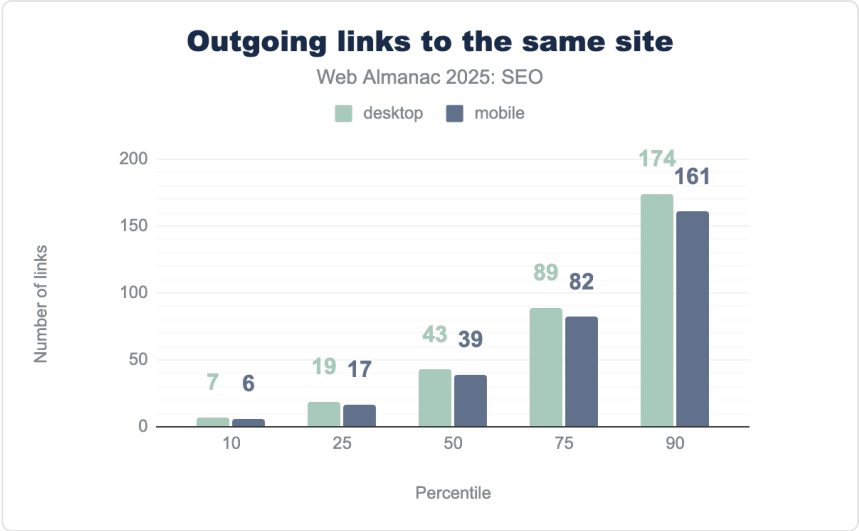


図5.41. 同一サイトへの発信リンク

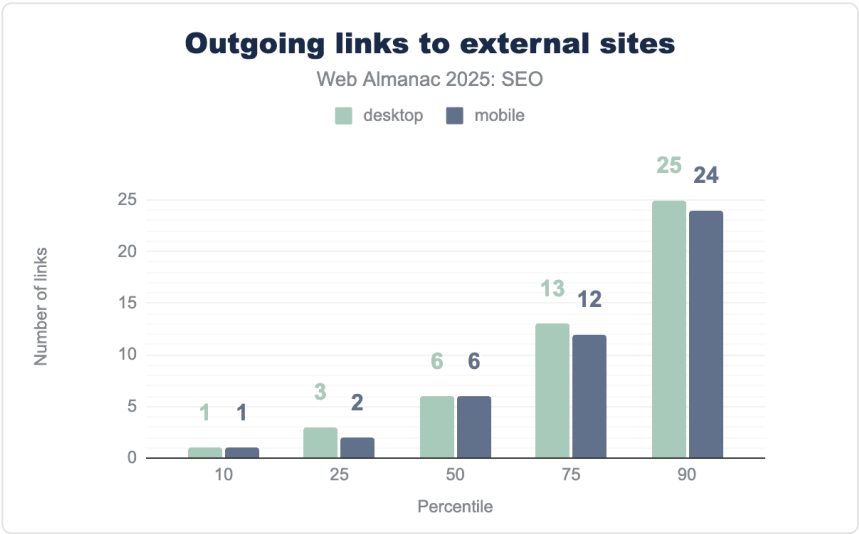


図5.42. 外部サイトへの発信リンク

アンカーの `rel` 属性の使用

`rel` 属性はもともと検索エンジンのためにリンクを修飾するためのものでしたが、今日最も一般的な使用はセキュリティに焦点を当てています。2025年、noopenerはページの40.9%

に、noreferrerは25%に表示されます。どちらもランキングには影響しませんが、タブハイジャックを防いで参照データを隠すことでサイトの健全性を高めユーザーを保護するため、広く採用されています。

検索関連の修飾子については、`nofollow`が32.3%のページで引き続き優位を占め、2024年の水準とほぼ同じです。対照的に、より特定的な`sponsored`と`ugc`属性はそれぞれ0.5%で停滞したままです。Googleが細かい分類を推進する取り組みにもかかわらず、採用は動いておらず、ウェブマスターが「`nofollow`対通常リンク」の大まかな区別を超えた価値をほとんど見出していないことを示唆しています。

総合すると、データはrel属性の使用パターンがほぼ安定していることを示しており、前年比の動きはほとんどありません。セキュリティプラクティスが使用をリードし、SEO修飾子は安定したままで、実験的な属性は勢いを示しません。dofollowやfollowなどの古い属性でさえ、0.5%未満のページにとどまっており、古いSEOの神話がウェブの小さな隅で実際の影響なく反響し続けていることを思い起こさせます。

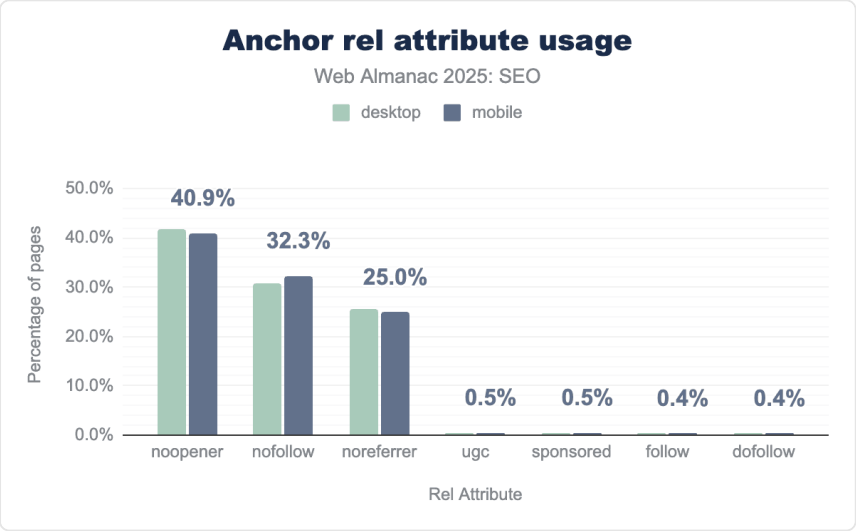


図5.43. アンカー`rel`属性の使用

ワード数

ワード数はSEOで長い間議論されてきた点で、コンテンツの品質や徹底さの代理指標として扱われることが多くあります。それにもかかわらず、ページの適切な長さについてのコンセンサスはほとんどありません。このセクションでは、生のページとレンダリングされたページにわたるワード数の分布を検討します。

レンダリングされたワード数

レンダリングされたワード数とは、JavaScriptが実行された後のページ上の可視の単語数を指します。

ホームページのレンダリングされたワード数

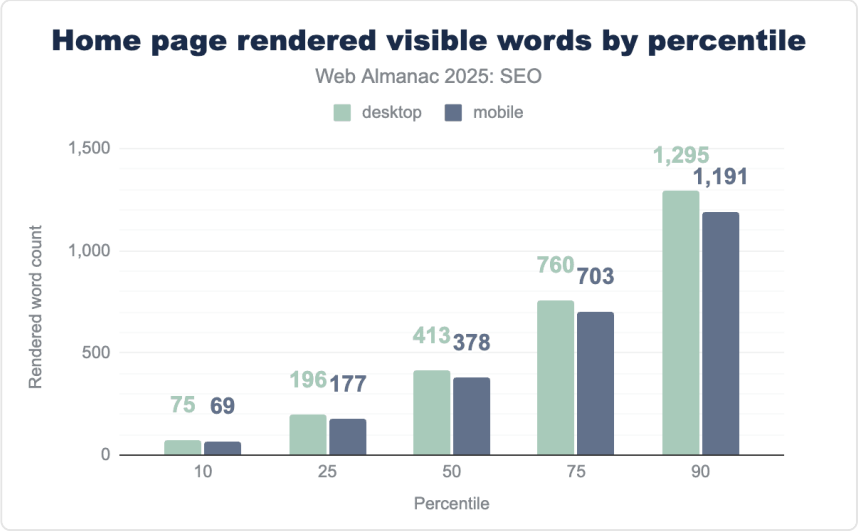


図5.44. パーセンタイル別のホームページのレンダリングされた可視単語数

2025年のモバイルホームページの中央値は378単語を含んでおり、2024年の中央値364単語からわずかに増加しています。デスクトップホームページのレンダリングされた中央値ワード数も前年比でわずかに増加しており、2025年は413単語（2024年の400単語から増加）を示しています。 モバイルとデスクトップのレンダリングされたワード数は2022年以降、デスクトップ中央値421単語、モバイル366単語だったものが低下を見せています。2025年のデスクトップとモバイルのパリティは昨年と比較してわずかに近づいており、モバイルとデスクトップのホームページワード数の差がわずか35単語に縮小し、デスクトップとモバイルのパリティが36単語だった2024年のレポートから1減少しています。これは年間を通じての一定の一貫性を示しています。

インナーページのレンダリングされたワード数

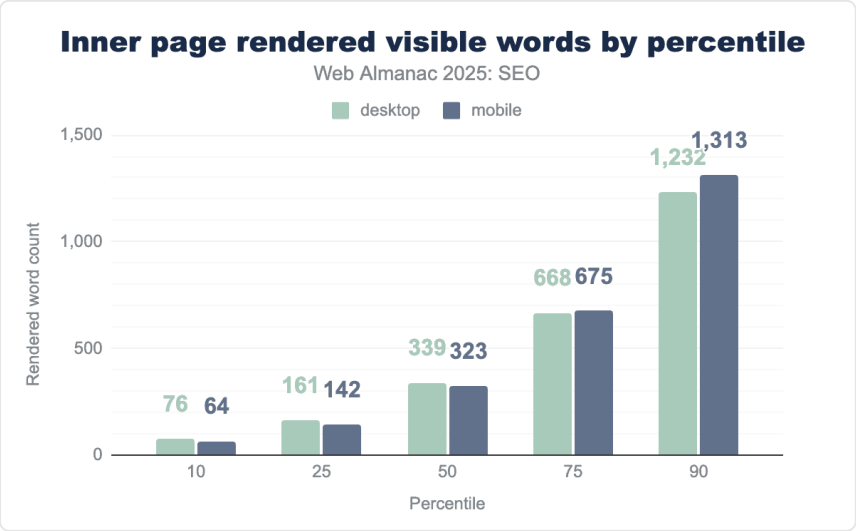


図5.45. パーセンタイル別のインナーページのレンダリングされた可視単語数

インナーページはホームページと比較して単語数が少ない傾向があります。2025年のモバイルインナーページの中央値はデスクトップで339単語、モバイルで323単語でした（デスクトップの中央値が333、モバイルが317だった2024年から増加しています）。

ここでのパーセンタイルは、インナーページの全データセットに対するページのワード数の位置を表しています。たとえば、90パーセンタイルはテストされたページの90%より多くの単語を持つページを反映しています。興味深いことに、50パーセンタイルまでは、デスクトップのインナーページが一般的にモバイルのインナーページより多くの単語を持つ傾向がありますが、パーセンタイルが高くなるにつれて差が縮まります。90パーセンタイルでは、デスクトップと比較してモバイルのインナーページワード数が大幅に増加しています。このトレンドは、デスクトップが常に高いワード数を持つホームページコンテンツでは見られません。

生のワード数

生のワード数とは、JavaScriptの実行またはDOMやCSSOMへのその後の変更の前に、サーバーから配信される元のHTMLに含まれる単語の総数を指します。

ホームページの生のワード数

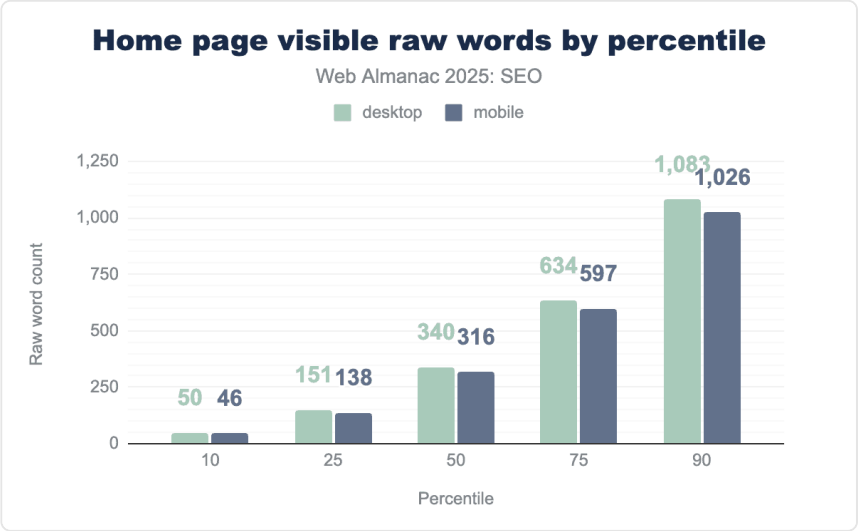


図5.46. パーセンタイル別のホームページの可視生単語数

2025年のページの生のワード数の中央値は、デスクトップユーザーエージェントで340単語、モバイルで316単語と、デスクトップで330単語、モバイルで311単語だった2024年からわずかに増加しています。

インナーページの生のワード数

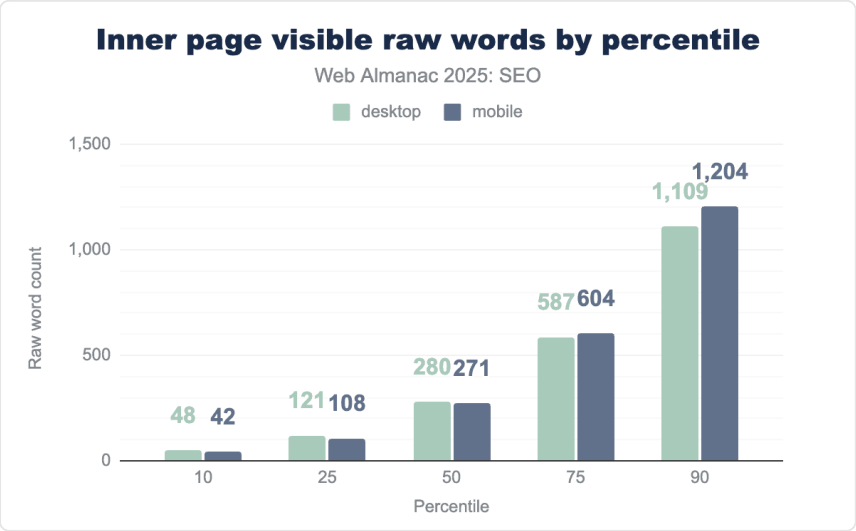


図5.47. パーセンタイル別のインナーページの可視生単語数

どのパーセンタイルでもモバイルがデスクトップより多くの単語を含まなかったホームページの生のワード数とは異なり、インナーページの可視ワード数は75パーセンタイル以上でデスクトップページがモバイルページより少ない単語数を示しています。

レンダリングされたワード数と生のワード数

ホームページでレンダリングされたワード数と生のワード数を比較すると、差異は小さく、中央値ではデスクトップで18%、モバイルで16%の差を示しています。これはモバイルで成長が見られており、2024年はわずか14%でした。

すべてのパーセンタイルにわたって、デスクトップユーザーエージェントに提供されるホームページは、モバイルと比較してレンダリング後に可視化される単語の割合が高くなっています。これはモバイルでのアコーディオンのような、DOMにレンダリングされていてもコンテンツが非表示になる一般的なレイアウトによるものと思われます。

モバイルとデスクトップのすべてのパーセンタイルにわたる最大の差異はわずか2%で、10パーセンタイルのような低いパーセンタイルでは等しいワード数を示しています。

ホームページのレンダリングされたワード数と生のワード数

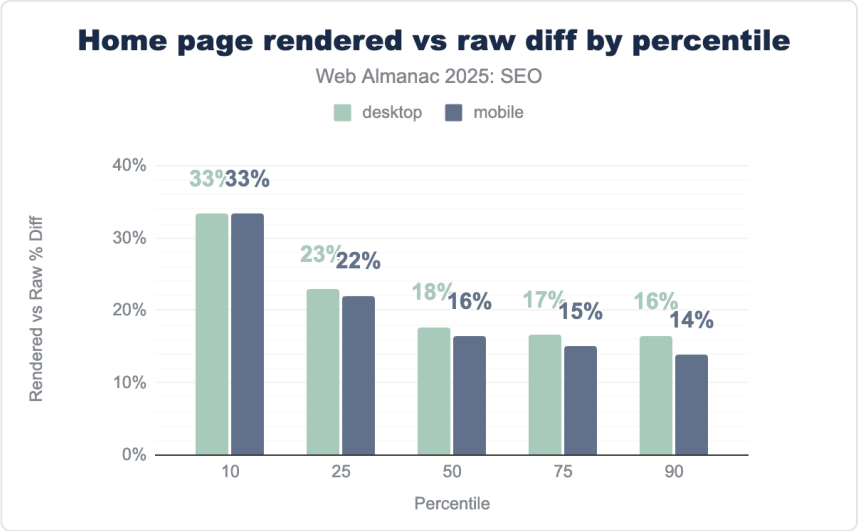


図5.48. パーセンタイル別のホームページのレンダリングと生の差分

2025年、10パーセンタイルでデスクトップ37%、モバイル34%と大幅な上昇が見られ、2024年のデスクトップ28%、モバイル22%と比較しています。

インナーページのレンダリングされたワード数と生のワード数

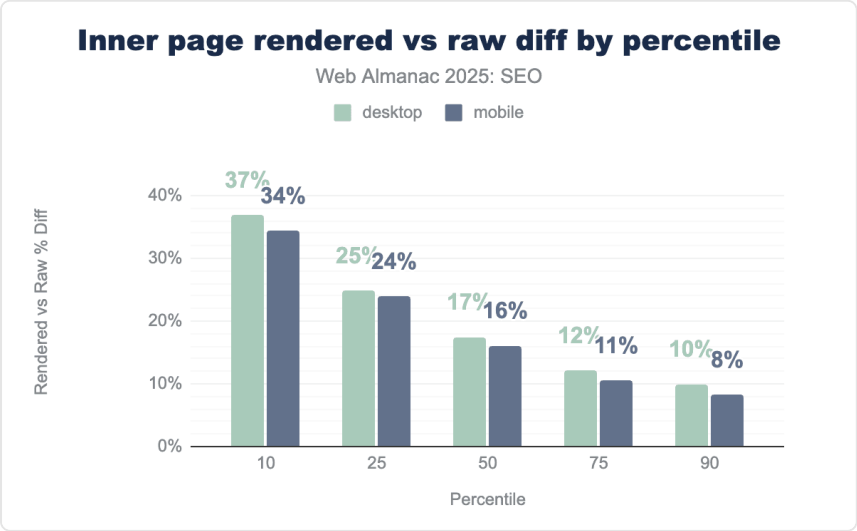


図5.49. パーセンタイル別のインナーページのレンダリングと生の差分

インナーページのレンダリングされたワード数と生のワード数の差は10パーセンタイルで最大のギャップが見られ、アコーディオンやタブレイアウトなどのモバイルブレークポイントでHTMLには存在するがモバイルでは非表示になっているコンテンツへの依存によるものと思われます。25、50、75パーセンタイルではわずか1%の差でギャップが縮まり、90パーセンタイルでは2%となります。

このパターンは単語が多いほど、クライアントサイドのレンダリングへの依存が少ないようで、これは2024年のチャプターでも同様でした。

構造化データ

直接のランキング要因ではないものの、構造化データは検索エンジンがページのコンテキストを理解するのを助け、検索エンジン結果ページ（SERP）でリッチリザルトを表示するための強力な手段として引き続き重要です。

構造化データ[136]は、大規模言語モデルがページの「内容」だけでなく「意味」を理解するためにも活用されるようになっています。MicrosoftはBingがschema.orgマークアップ[137]を使用して、専門家記事・製品・レビュー・FAQをBing ChatやCopilotが区別できるようにしてい

136. https://insightland.org/blog/structured-data-ai-search/
137. http://schema.org

ることを認めています。GoogleのAIオーバービュー[138]の動作も同様の依存を示唆しており、GPTBotなどのクローラーは状況によってはHTML内に埋め込まれたスキーマを解析できます。

ホームページの構造化データ

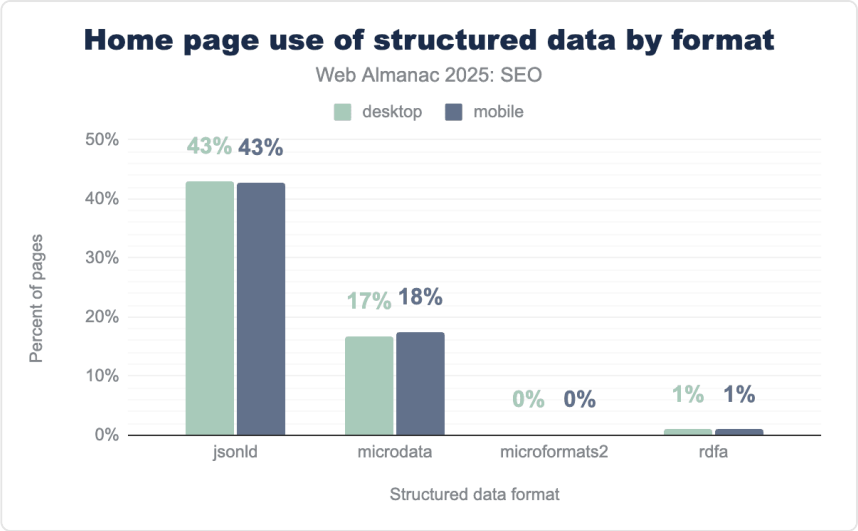


図5.50. Home page use of structured data by format

構造化データの全体的な使用率は、2024年のデスクトップ48%・モバイル49%と比べ、2025年はデスクトップ・モバイルともに50%に成長しました。

大半のサイトは生のHTMLで構造化データを提供しており、JavaScriptで追加するケースはモバイル・デスクトップともに2%のみ（2024年から変化なし）です。

GoogleはJavaScriptで注入されたスキーマも処理できますが、初期HTMLに含めることで処理の遅延・クロールの問題・ユーザー体験の低下を防ぐことができます。

JSON-LDはホームページで圧倒的に人気のフォーマットで、2024年のデスクトップ41%・モバイル40%と比べ、2025年はデスクトップ・モバイルともに43%を占めています。

Googleは通常、JSON-LD構造化データの使用を推奨しています[139]。実装・保守が最も簡単なためですが、MicrodataやRDFaなど他のフォーマットもクロール・解析は可能です。Microdataの使用率はデスクトップ・モバイルともに2024年比で1%低下しています。

138. *https://www.semrush.com/blog/how-can-schema-markup-specifically-enhance-llm-visibility/*
139. *https://developers.google.com/search/docs/appearance/structured-data/intro-structured-data*

インナーページの構造化データ

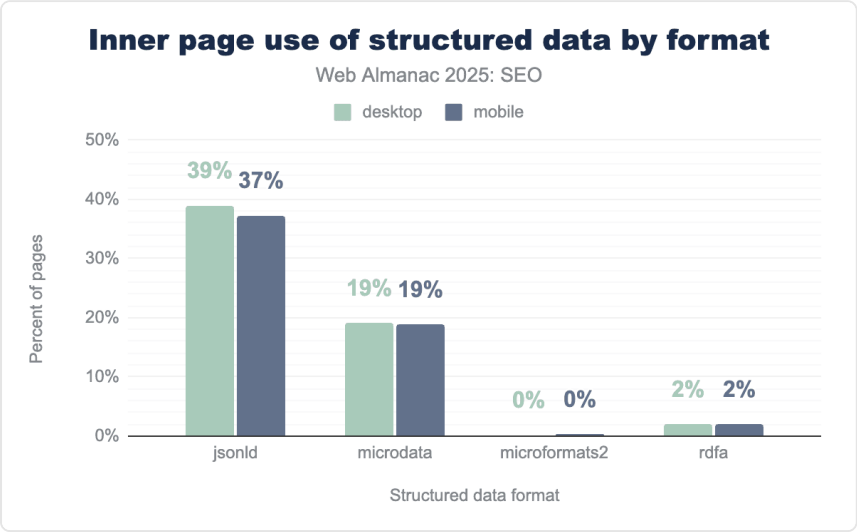


図5.51. Inner page use of structured data by format

インナーページでは若干の差異が見られ、Microdataはモバイル・デスクトップともに19%、JSON-LDはデスクトップ39%に対してモバイルは37%となっています。

これはホームページが最も多くのSEO対策を受けており、OrganizationやWebsiteスキーマなどのグローバルサイトテンプレートを通じて、手動またはJSON-LDで直接実装される可能性が高いためかもしれません。ホームページはリッチリザルトやブランドナレッジパネルで優先されることが多いため、JSON-LDの採用率はデバイス間でより高く一貫している傾向があります。

一方、製品ページや記事ページなどのインナーページは、デスクトップとモバイルのテンプレートによって異なる動的またはCMS生成マークアップに依存することが多く、若干の差異が生じます。

最も人気の高い構造化データタイプ

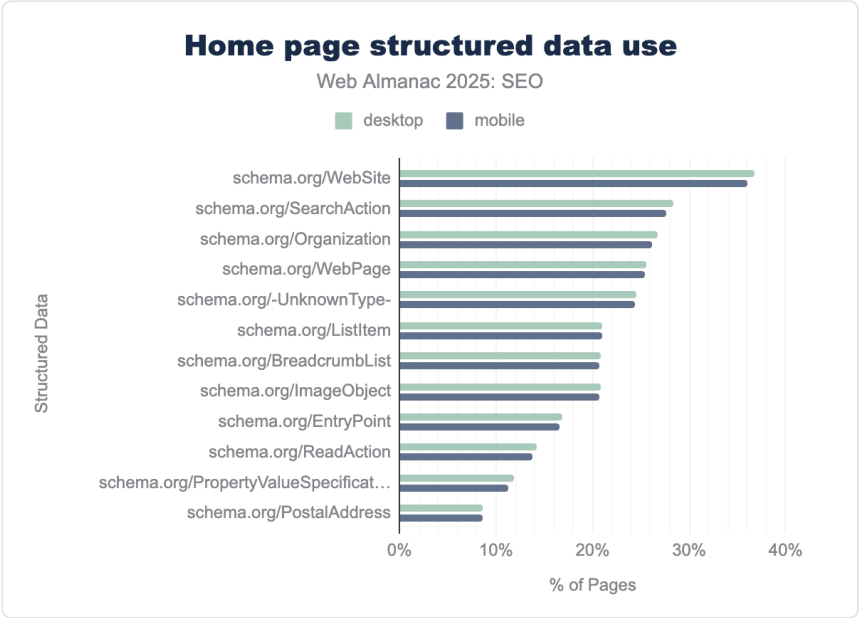


図5.52. ホームページの構造化データ使用状況

最も人気の高い構造化データタイプ上位2位に大きな変化はなく、2022年・2024年・2025年を通じてWebSiteとSearchActionが最も人気です。これらはGoogleのサイトリンク検索ボックスを機能させるものですが、視覚的には2024年11月に廃止されました。ただし、それを支える構造化データを削除する必要はありません。

Organizationスキーマの人気が若干上昇し、2025年のデータでは初めてWebPage スキーマ（Web**Site**スキーマと混同しないでください）を上回りました。

WebSiteスキーマは引き続きすべての構造化データタイプの中で最も人気で、2025年のモバイルでは2024年の35%・2022年の30%と比べ36%に若干増加しました。モバイルの構造化データ使用率は年ごとの差異が小さいです。

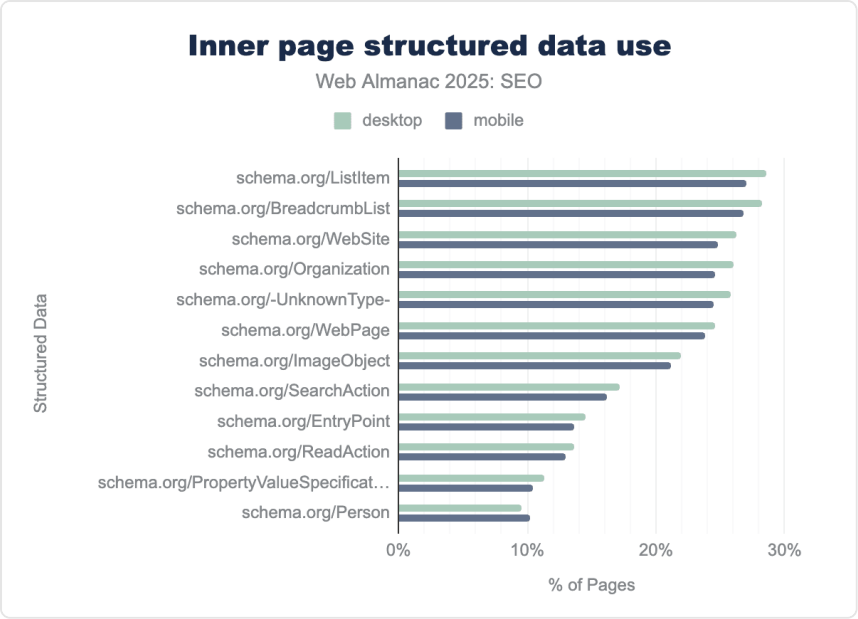


図5.53. インナーページの構造化データ使用状況

インナーページでは、ListItem構造化データが引き続き最も人気のインナーページスキーマタイプですが、モバイルの使用率は2024年の30%から2025年の27%に低下しています。

興味深いことに、WebSite構造化データはホームページ固有のスキーマタイプであるにもかかわらず（少なくともGoogleによれば[140]）、インナーページでも3番目に人気の構造化データタイプです。これは多くの種類のスキーマがCMSレベルでテンプレート化されており、インナーページにも複製されているためかもしれません。

スキーマの使用状況の詳細については、専用の構造化データチャプターをご覧ください。

AMP

Accelerated Mobile Pages（AMP）はかつて、特にモバイル検索におけるパフォーマンスと視認性を高める近道として推進されていました。しかし、すべてのページに直接パフォーマンス指標を提供するCore Web Vitalsが登場したことで、AMPの役割は急激に縮小しました。2025年のデータでは、ホームページ・インナーページを通じて採用率は一貫して低く、1%を下回るままです。残存する使用はニッチなまたはレガシーな実装を反映してお

140. *https://developers.google.com/search/docs/appearance/structured-data/organization?_gl=1*jf94bx*_up*MQ..*_ga*OTY2MjgzMjIwLjE3NjIzMjkyODU.*_ga_SM8HXJ53K2*czE3NjIzMjkyODQkbzEkZzAkdDE3NjIzMjkzMDIkajQyJGwwJGgw#guidelines*

り、主流の戦略とはなっていません。

ホームページのAMP使用状況

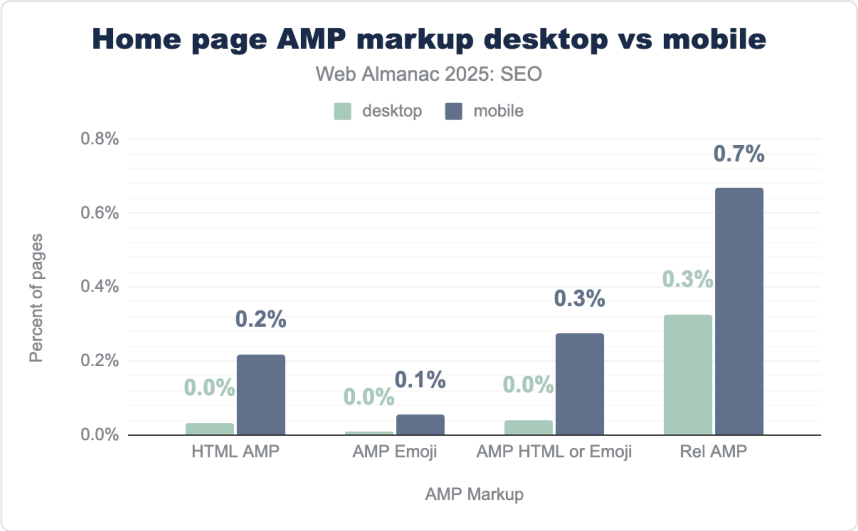


図5.54. ホームページのAMPマークアップ：デスクトップ対モバイル

AMPマークアップの採用は2025年も極めて低い水準にとどまり、インナーページではすべてのデバイスタイプでわずかな使用しか見られません。`rel=amphtml` が最も一般的なシグナルで、デスクトップページの0.8%、モバイルページの0.9%に出現しています。HTML AMPはモバイルページのわずか0.1%に見られ、デスクトップではほぼゼロです。AMP EmojiやAMP HTML + Emojiなど他のバリアントはほぼ存在せず、0.2%以下です。

2024年と比較すると、全体的な状況は停滞と言えます。2024年に見られた若干の上昇は継続せず、使用率は引き続き1%を大きく下回ったままです。GoogleがTop Storiesの要件からAMPを外したことと、Core Web VitalsがユニバーサルなパフォーマンスメトリクスとしてAMPの地位を確実にフリンジテクノロジーに押し込んでいます。

ホームページ対インナーページのAMP使用状況

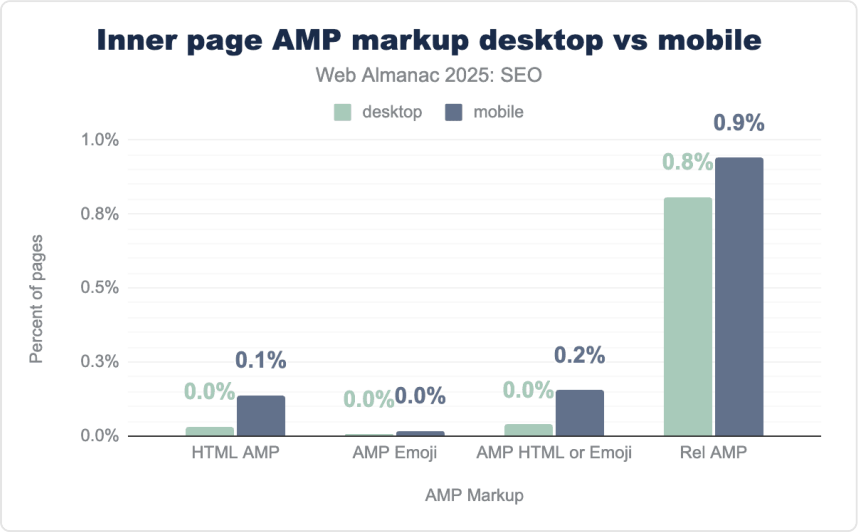


図5.55. インナーページのAMPマークアップ：デスクトップ対モバイル

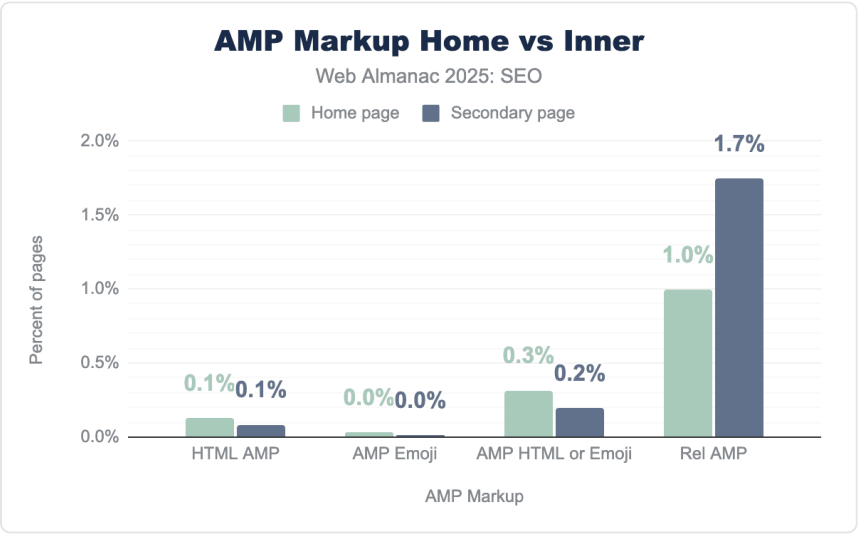


図5.56. AMPマークアップ：ホームページ対インナーページ

2024年はホームページがインナーページよりわずかにAMPを採用している傾向がありましたが、2025年はその逆となっています。`rel=amphtml` はインナーページの1.7%に出現するのに対し、ホームページは1.0%です。HTML AMPも同様で、採用率は現在均等（各

0.1%）に分かれています。

国際化

hreflang属性は国際サイトや多言語ウェブサイトにとって重要な機能であり、検索エンジンがユーザーに適切な言語や地域バージョンのページを提供するのを助けます。2025年、hreflangはおよそ20%のページに出現しています。採用は国際ターゲティングが最も大きな影響を持つホームページで優先されることが多く、インナーページはカバレッジが一貫していません。成長は継続していますが、広く使用される少数の言語に集中しています。

`hreflang` の実装

現在、5ページに1ページがhreflangマークアップを含んでいます。生（デスクトップ）は20.3%、生（モバイル）は19.7%で、レンダリング後はそれぞれ20.8%・20.2%とわずかに高い値です。これはhreflangタグを持つページが9〜10%程度だった昨年の数値のほぼ2倍です。

HTTP hreflangはほぼ消滅しており、デスクトップページの0.2%・モバイルページの0.1%が使用しているのみです。これは2024年のすでに小さなシェアと一致しており、HTTPSが国際化ウェブサイトのほぼすべてでデフォルトとなったことを裏付けています。

生とレンダリング後の値の近似性（1%未満の差）は、hreflangを実装しているほとんどのサイトが技術的に正しい方法で実装しており、レンダリング後も維持されていることを示しています。これはギャップがわずかに大きかった2024年からの改善です。

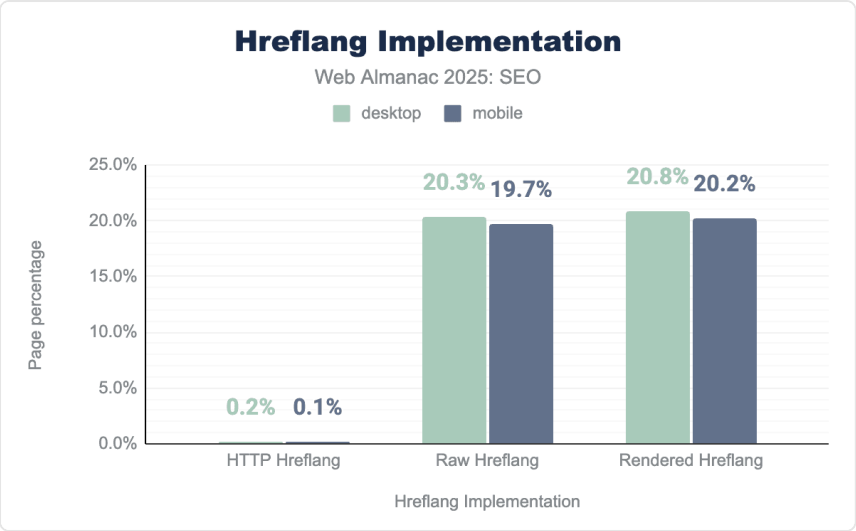


図5.57. `hreflang` の実装方法

ホームページの `hreflang` 使用状況

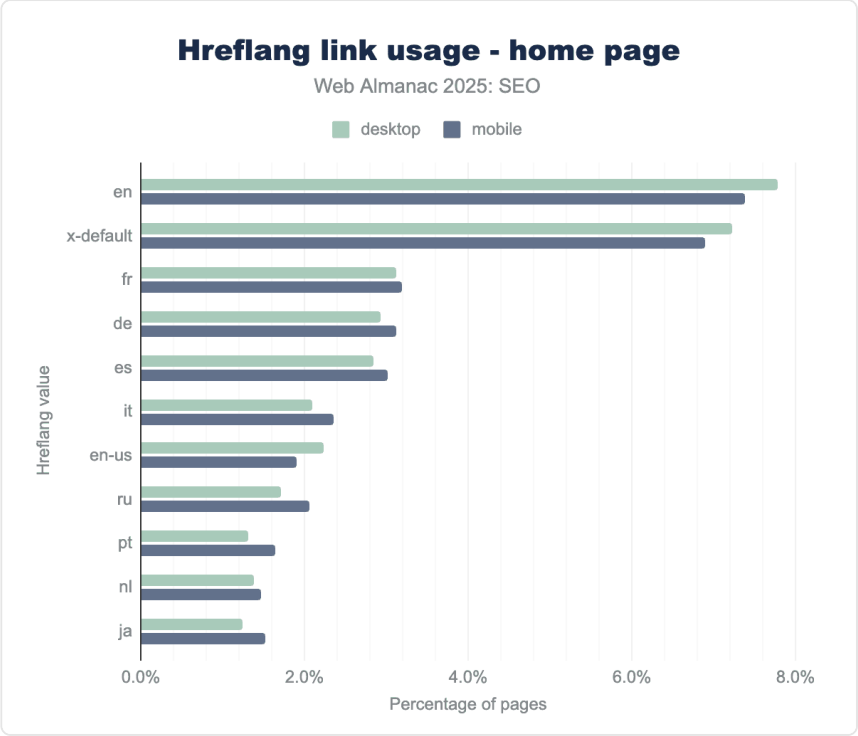


図5.58. hreflang リンクの使用状況 - ホームページ

ホームページでの `hreflang` 使用は依然として少数の値に高度に集中しています。英語（`en`）が引き続き支配的で、デスクトップホームページの7.4%、モバイルの7.3%に出現し、`x-default` がデスクトップ7.2%・モバイル6.9%で続いています。

フランス語（モバイル3.2%）・ドイツ語（モバイル3.1%）・スペイン語（モバイル3.0%）などの第2言語グループが次のティアを形成し、イタリア語・ロシア語・ポルトガル語・オランダ語・日本語は2.5%未満にとどまっています。

この集中は、`hreflang` の採用がページ全体の20%をカバーするまでに成長したにもかかわらず、その大半が広くターゲットとされる少数の視聴者層に集中していることを示しています。

2024年に `en` がデスクトップ・モバイルともにホームページの8%に出現していたのと比べ、2025年のデータは若干の低下を示しています。ただし、第2言語群の採用は安定しており、全体的な分布は年を通じて一貫しています。

インナーページの `hreflang` 使用状況

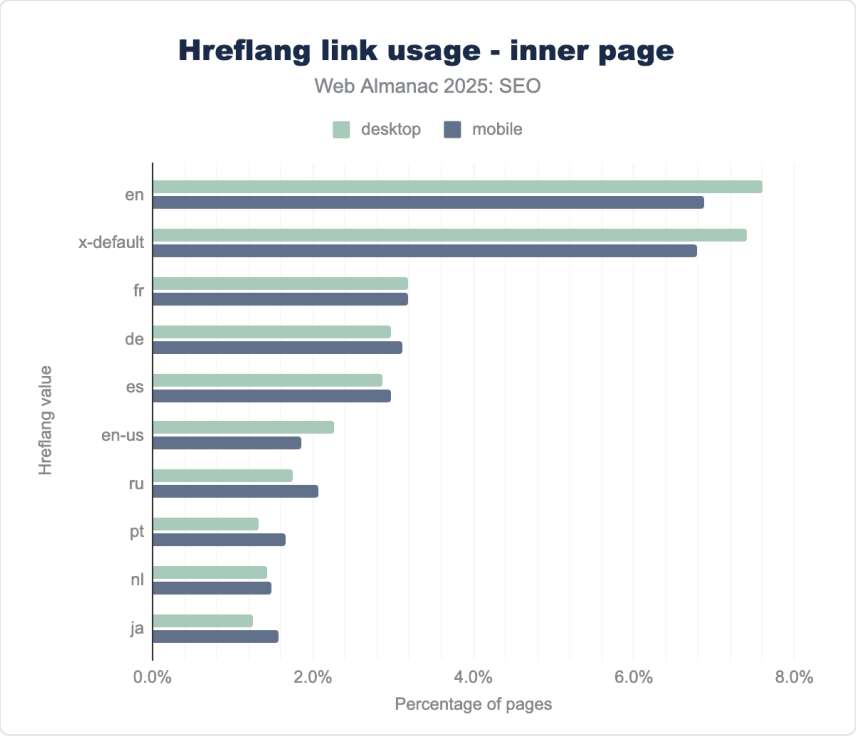


図5.59. hreflangリンクの使用状況 - インナーページ

インナーページでのhreflang採用はホームページのパターンを反映していますが、わずかに低い水準です。英語（en）がデスクトップページの7.6%・モバイルの6.9%でトップを占め、x-defaultがデスクトップ7.4%・モバイル6.8%で続いています。フランス語（インナーモバイルページの3.2%）・ドイツ語（モバイル3.1%）・スペイン語（モバイル3.0%）が次のティアを形成し、他のすべての言語値は2.5%未満にとどまっています。

2024年にデスクトップのx-defaultとenが8%近くだったのと比べ、2025年の数値はわずかな低下を示しています。この分布は、hreflangがインナーページよりもホームページで優先される傾向にあることを再確認するものです。フランス語・ドイツ語・スペイン語の緊密なクラスタリング（インナーモバイルページで3.0〜3.2%）は、英語以外の多言語ウェブ採用の大部分が欧州主要市場によって推進されていることを裏付けています。

結論

AIサーチがコンテンツの発見方法を再形成する中、ウェブの基本原則はかつてないほど重要です。心強いことに、データはその基盤が揺るぎないことを示しています。

クロール可能性とインデックス可能性は依然として強固です。ほとんどのサイトが有効な `robots.txt` ファイルと `canonical` タグを提供しており、マークアップの品質は年々改善しています。`robots.txt` 自体の役割も進化しており、アクセスのゲートキーパーからコンテンツ使用意図の宣言へと移行しています。数百万のファイルが `gptbot` や `claudebot` などのAIクローラーに明示的に対応するようになっています。強固な `meta` ロボットの採用と安定したカノニカルシグナルと合わせて、これらの技術的な柱は人間とマシン主導の検索システムの両方にとって信頼できる基盤を提供しています。

これらのアクセスシグナルを超えて、ウェブの中間層——ページエクスペリエンス・パフォーマンス・オンページ構造——が心強い安定性を示しています。HTTPSの採用は91%を超え、Core Web Vitalsは歴史的に高い水準で横ばいとなり、タイトル・ディスクリプション・見出しなどのオンページメタデータは検索とAI主導のインデックス作成の両方に向けて最適化が進んでいます。構造化データの採用率は50%に達し、ほとんどの実装は直接HTMLに埋め込まれています。これらはすべて、クローラーとモデルによる、より速く・よりクリーンな解釈を確保するためのものです。

`llms.txt` はウェブがAIシステムに直接語りかけるための最新の実験です。サイトオーナーは、コンテンツがAIによってどのようにランク付けされ・読まれ・再利用されるかについて考えるよう招かれています。採用率はまだ低く（約2%）、意図的な戦略よりもCMSプラグインによって推進されていますが、その存在はマインドセットの転換を示しています。`llms.txt` が標準になるか脚注にとどまるかは、2026年に持ち越される重要な問いです。

この進歩の多くは、ベストプラクティスをデフォルトで組み込むようになったCMSプラットフォームから生まれており、ウェブ全体の技術的なベースラインを静かに引き上げています。テクニカルSEOの全体的な軌跡は心強いものです。ウェブは1年前よりも一般的により一貫性があり・クロールしやすく・意味的に高度になっています。

SEO担当者は長年、これらの基本原則——構造化データ・メタデータ・クリーンなアーキテクチャ——が重要であることを知っていましたが、絶え間ないアルゴリズムの変動が、しばしばそれらを手っ取り早い戦術よりも二次的なものと感じさせていました。しかし今、大規模言語モデルがコンテンツの解釈と引用のために構造化され適切にラベル付けされたデータにますます依存するようになった今、それらの同じ基本原則が再び実証されています。基礎は時代遅れになったことはなく、今後もそうなることはないでしょう。

著者

Amaka Chulwuma

Amaka-maxi amaka-chukwuma-jubilee

AmakaはSEOおよびコンテンツストラテジストで、過去7年間にわたりブランドのオンラインプレゼンスを形成してきました。英国、米国、オーストラリアのエージェンシー（Whitecoat SEOやSwitch Key Digitalを含む）で勤務し、法律、医療、家庭サービス、B2B分野のクライアントのためにコンテンツシステム、テクニカルSEOの基盤、検索主導のストーリーテリングを構築しています。その仕事は明確さ、思慮深い戦略、共感のミックスを反映しています。仕事以外でも同じ姿勢を持ち込んでおり、特に毎日の最良の部分である娘との時間を大切にしています。

Chris Green

@chrisgreenseo @chris-green.net chr156r33n chrisgreenseo

https://www.torquepartnership.com/

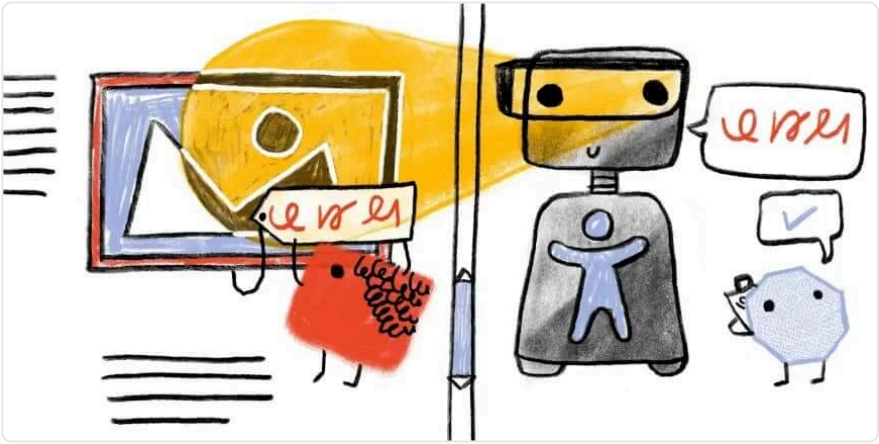
Chris GreenはTorque Partnershipのテクニカルディレクターであり、15年以上のサーチ業界のベテランです。Fortune 500企業に対して、検索戦略やブランド・アルゴリズム・AIシステムの進化する関係性についてアドバイスしています。

Sophie Brannon

@SophieBrannon SophieBrannon

Sophie BrannonはアトランタをStudioHawk USの共同創設者兼ディレクターです。エージェンシー、社内、コンサルタントにわたる10年以上のSEO経験を持ち、eコマース、金融、ゲーム、ヘルス、SaaSなど幅広い業界で活動してきました。Sophieはオーガニック成長に対して戦略的かつデータドリブンなアプローチを採用し、テクニカルSEO、ユーザーエクスペリエンステスト、コンテンツ、デジタルPRを通じてブランドの持続可能なスケールを支援しています。SEOをアクセスしやすくし、次世代の検索マーケターを育成することに情熱を持っています。

部 II 章 6
アクセシビリティ



Bogdan Lazar と Mike Gifford によって書かれた。
Hidde de Vries、 Aidan Tierney と Scott Davis によってレビュ。
Mike Gifford と Barry Pollard による分析。
Bogdan Lazar 編集。
Sakae Kotaro によって翻訳された。

はじめに

ウェブは急速に変化しています。2025年、主流のテクノロジーがインクルーシブな機能にますます依存するようになる中、ウェブアクセシビリティはかつてないほど重要です。例えば、音声アシスタントはスクリーンリーダー技術を使用しています。動画キャプションや触覚フィードバックなど、もともとアクセシビリティのために設計された機能は今では一般的になっています。

ユニバーサルデザインの原則は現代のウェブ開発において根本的に重要です。私たちは多様なニーズに対応し、すべてのユーザーの体験を改善するソリューションをますます作り出しています。Tim Berners-Lee卿が有名な言葉を残しているように、「ウェブの力はその普遍性にある。障害の有無にかかわらず誰もがアクセスできることは必須の側面だ。」

最近の世界的な出来事と変化する法的要件が、デジタルインクルージョンを注目の的にして

います。MicrosoftのInclusive Design Guidelines[141]は、アクセシビリティが永続的な障害を持つ人々だけでなく、より多くの人々を助けることを示しています。このガイドラインは一時的・状況的な制限についても具体的に言及しています。例えば、片手でデバイスを使う能力は、怪我をした人、幼い子どもを持つ親、荷物を持っている人にも役立ちます。

2025年、ウェブアクセシビリティ法には実質的な効力があります。欧州連合（EU）の欧州アクセシビリティ法（EAA）[142]は大きな前進です。ウェブコンテンツアクセシビリティガイドライン（WCAG）[143]を参照するEN 301 549[144]標準への準拠について、多数のウェブサイトとアプリに対して2025年6月を期限として設定しました。

米国も規制を更新しました。州・地方政府のサイトは、米国障害者法タイトルIIで義務付けられた[145]通り、WCAG 2.1[146]を満たす必要があります。2024年のデータは、これらの期限がウェブサイトのアクセシビリティに与える具体的な影響を測定するための重要なベースラインを提供しています。

分析にはDequeのaxeコアエンジンによるGoogle Lighthouse[147]を使用しています。2025年の結果を2024年のデータと比較し、主要なトレンドを特定します。WCAG 2.2のより広い採用に伴い、新しい達成基準の普及と `duplicate-id` などの廃止ルールからの継続的な変化を検討します。

私たちのアプローチはWebAim Million[148]と似ていますが、クロールするサイトと使用する分析ツールが異なります。HTTP Archiveは毎月Lighthouseやその他のツールを使用して、ホームページとセカンダリページにわたって1,700万サイトをクロールしています。WebAimはWAVE[149]で上位100万ホームページ[150]を調査しています。

LighthouseにもaXeコアを使った自動テストは、WCAG達成基準のサブセットを部分的にしか確認できません[151]。GOV.UKのAlphagovは一般的な自動監査ツールの比較[152]を提供していますが、すべてのツールがアクセシビリティエラーの50%未満しか検出していません。多くの基準には自動テストがなく、すべてのアクセシビリティ問題がWCAGの基準に対応しているわけでもありません。

グッドハートの法則を覚えておいてください。指標がターゲットになると、信頼できる指標ではなくなります。完璧なスコアは完全なアクセシビリティを保証しません。Lighthouseのアクセシビリティスコアは最終目標ではなく、評価の出発点として扱うべきです。それでも、これらのスコアを追跡することはウェブの全体的な進歩の貴重なスナップショットを提供します。

141. *https://inclusive.microsoft.design/*
142. *https://en.wikipedia.org/wiki/European_Accessibility_Act*
143. *https://www.w3.org/WAI/standards-guidelines/wcag/*
144. *https://en.wikipedia.org/wiki/EN_301_549*
145. *https://www.ada.gov/resources/2024-03-08-web-rule/*
146. *https://www.w3.org/TR/WCAG21/*
147. *https://www.deque.com/axe/axe-core/*
148. *https://webaim.org/projects/million/*
149. *https://wave.webaim.org/*
150. *https://webaim.org/projects/million/#method*
151. *https://html5accessibility.com/stuff/2025/03/24/a-tools-errand/#:~:text=automation%20blues*
152. *https://alphagov.github.io/accessibility-tool-audit/*

私たちのレポートはHTMLのみに焦点を当てており、PDFやその他のオフィスドキュメントは含みません。

2024年と比較して、Lighthouseアクセシビリティスコアの中央値は1%改善し、2025年には85%を超えました。2019年の最初のWeb Almanac以来、着実で段階的な進歩が見られています。Google LighthouseはaxeコアのAの問題に異なる重みを割り当てています[153]ので、組織によって修正の優先順位が異なる場合があります[154]。

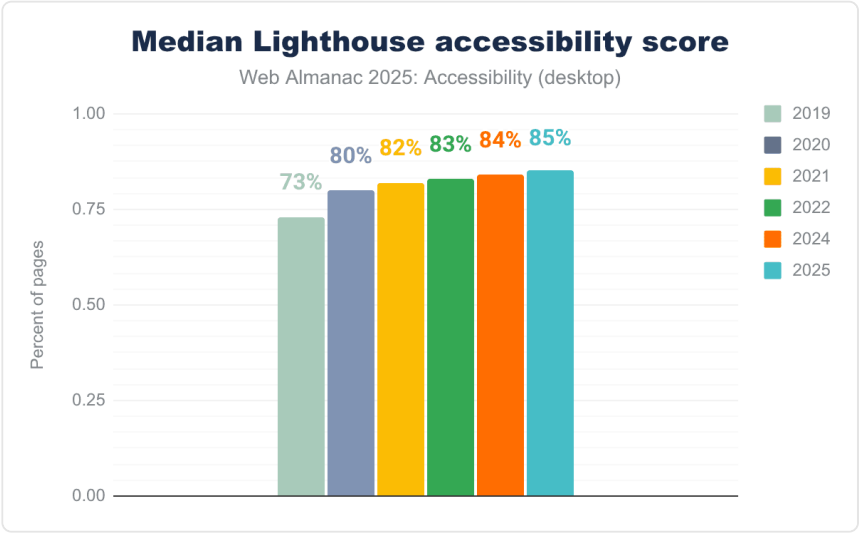


図6.1. Lighthouseの監査改善状況（前年比）。

今年、以下のaxeコアテストで最大の進歩が見られました：

- ARIAの入力フィールドにはアクセシブルな名前が必要[155]：2024年比 +3%
- ARIAの `meter` ノードにはアクセシブルな名前が必要：2024年比 +15%
- ARIAの `progressbar` ノードにはアクセシブルな名前が必要：2024年比 +5%
- ARIAの `tooltip` ノードにはアクセシブルな名前が必要：2024年比 +13%
- 20時間未満の遅延リフレッシュを避ける[156]：2024年比 +1%
- `<object>` 要素には代替テキストが必要：2024年比 +1%

153. *https://developer.chrome.com/docs/lighthouse/accessibility/scoring*
154. *https://accessibility.civicactions.com/posts/prioritizing-accessibility-bugs-for-maximum-impact*
155. *https://dequeuniversity.com/rules/axe/4.11/aria-input-field-name*
156. *https://dequeuniversity.com/rules/axe/4.11/meta-refresh*

- `<select>` 要素にはアクセシブルな名前が必要：2024年比 +5%

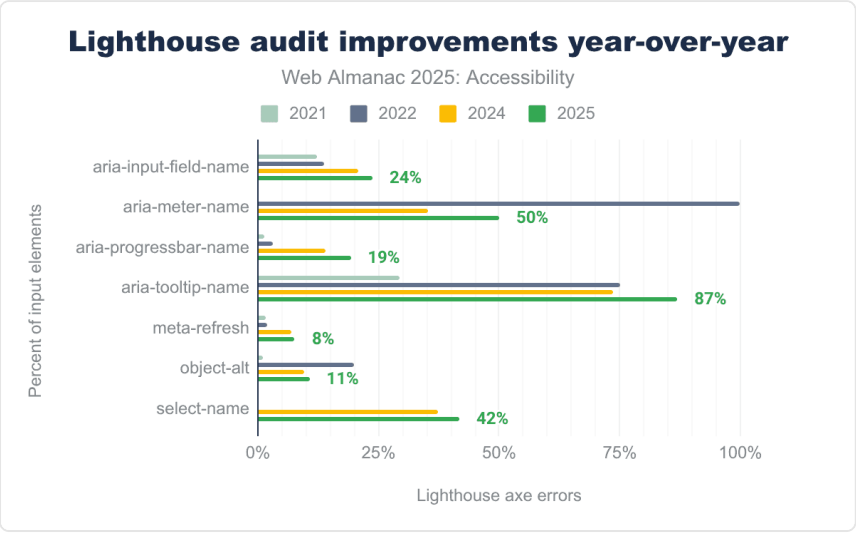


図6.2. 最も改善されたLighthouseアクセシビリティテスト（axe）。

今年、Web Almanacに人工知能（AI）に関する新しいセクションを追加しました。AIはウェブサイトの構築・コードの記述・コンテンツの生成・パフォーマンスの最適化・インターフェースとのインタラクション方法を変えています。画像の説明やキャプションの生成から、チームが問題を見つけ修正するのを助けるアシスタントまで、アクセシビリティ作業においても役割が増しています。

同時に、AIはリスクと未解決の問題をもたらします。ウェブサイトがAIによって作成されたり、AIの支援を受けたりした場合を確認する信頼できる方法はまだありません。言語モデルはアクセシビリティ問題を含むコードやコンテンツで訓練されています。自動化された説明やパターンはコンテキスト・ユーザーの意図・ニュアンスを見逃しやすいです。データ使用・環境への影響・内在するバイアスに関する広範な懸念は、ウェブ上のAIから誰が恩恵を受け、誰が傷つけられ排除されるかに直接影響します。

新しいAIチャプターはこれらの緊張を探求しています：AIがすでにチームをどのように助けているか、どこで不足しているか、どのような標準・保護措置・実践が必要か。分析全体を通じて一つの原則が貫かれています：AIは人間の専門知識とインクルーシブデザインを置き換えるのではなく、支援すべきです。

このチャプター全体を通じて、自分のサイトのアクセシビリティを改善するための実践的なリンクと具体的なソリューションを見つけることができます。

読みやすさ

ユーザーはウェブコンテンツを簡単に読んで理解できる必要があります。これは読みやすいフォントの選択にとどまりません。明確な言語の使用・ページの論理的な整理・予測可能なデザインパターンへの準拠も含まれます。

このレポートは測定可能な技術的指標に焦点を当てていますが、平易な言語での執筆などの定性的要素も同様に重要です。WCAG 3.0は明確な言語の要件を組み込む方法を検討していますが、まだ開発段階にあります。

平易な言語と同様に、数値もアクセシビリティ上の課題をもたらします。一部のユーザーは数値の解釈に苦労しており、自動テストでは障壁として確実に検出できません。これに対処するには、ウェブ上で数値情報を明確に提示するための実践的なアドバイスを提供するAccessible Numbers[157]などのリソースを参照してください。

英語のコンテンツには読みやすさの指標があります。Flesch-Kincaid[158]の可読性スコアはその一例です。しかしウェブはグローバルです。多くの言語と多様な視聴者にまたがっています。すべてのケースや言語をカバーする標準化された自動テストは存在しません。

カラーコントラスト

前景色と背景色の差異が、ユーザーがウェブコンテンツを認識できるかどうかを決定します。不十分なカラーコントラストは、特に低視力のユーザーや色覚異常[159]を持つユーザーにとって、依然として一般的な障壁です。

カラーコントラストは、高齢ユーザー・老眼鏡を忘れた人などの一時的な障害を持つ人・明るい日光や困難な環境でコンテンツを読む人にとっても特に重要です。

WCAGはAA適合性を達成するために、通常テキストに対して少なくとも4.5:1、大きなテキストに対して3:1のコントラスト比を要求しています。AAA適合性は通常テキストに対して7:1を要求します。WCAGのコントラスト比[160]は重要なベースラインですが、これらのガイドラインはすべての形態の色覚異常や個人の知覚の違いに対応しているわけではありません。

Accessible Perceptual Contrast Algorithm（APCA）[161]を含む他のドキュメントは、より知覚的に正確なコントラストの測定を提供することを目的としています。

オープンソースツール[162]（新しくリリースされたContrast Report[163]など）は、カラーコントラストの問題を見つけて修正することをこれまで以上に簡単にしています。必要な比率を満

157. *https://accessiblenumbers.com*
158. *https://en.wikipedia.org/wiki/Flesch%E2%80%93Kincaid_readability_tests*
159. *https://www.colourblindawareness.org/*
160. *https://developer.mozilla.org/docs/Web/Accessibility/Guides/Understanding_WCAG/Perceivable/Color_contrast*
161. *https://git.myndex.com/*
162. *https://accessibility.civicactions.com/guide/tools#color*
163. *https://contrast.report/*

たさない色に対して修正案も提示します。追加のガイダンスとして、Dennis Deaconのカラーコントラストテストに関する記事[164]などの専門家リソースを参照できます。

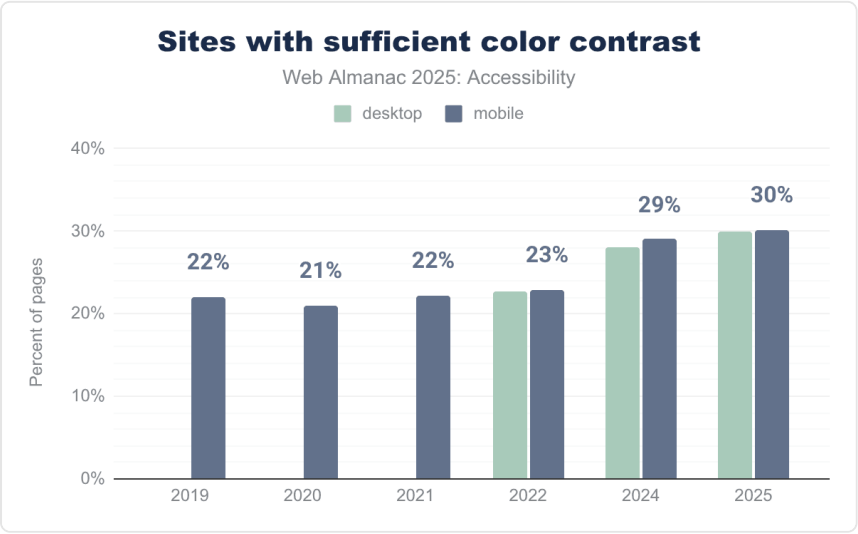


図6.3. 十分なカラーコントラストを持つサイト。

今年、テキストコントラストの合格率は2024年比でおよそ1%改善しました。しかし、現時点でモバイルサイトの31%のみが最低限のカラーコントラスト要件を満たしています。モバイル体験は明瞭な視認性に大きく依存しているため、このギャップはスマートフォンでウェブにアクセスするユーザーにとって深刻な問題です。

ブラウザとオペレーティングシステムはライト・ダーク・ハイコントラストモードのサポートを強化しています。ユーザーはより多くのコントロールを持つようになっています。しかし、ほとんどのサイトはまだこれらの設定に対応していません。

ズームとスケーリング

ユーザーはニーズに合わせてコンテンツをリサイズできる必要があります。ズームを無効にするとユーザーコントロールが失われ、WCAGのリサイズ要件に直接違反します。これは単なる小さな不便にとどまりません。低視力のユーザーや読書に画面拡大機能を使用している人にとって、サイトが完全に使用不能になる可能性があります。

2025年でも、この制限的なパターンはまだ見られます。多くの場合、開発者がモバイルデバイスでピクセルパーフェクトなレイアウトを望むためです。残念ながら、それはユーザビ

164. *https://www.dennisdeacon.com/web/accessibility/testing-methods-use-of-color/*

リティとアクセシビリティを犠牲にしています。

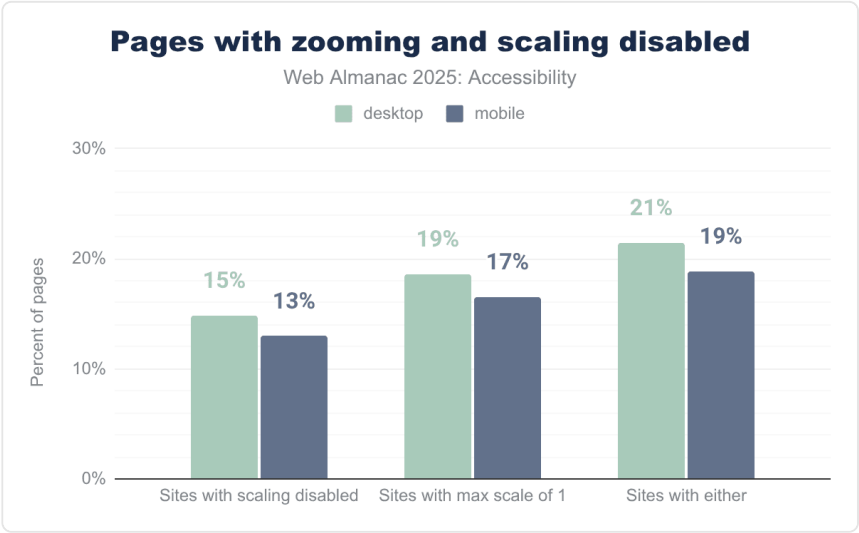


図6.4. ズームとスケーリングが無効になっているページ。

ズームやスケーリングを無効にするサイトの数は減少し続けています。2025年には、`user-scalable=no` の使用または制限的な最大スケールの設定によってスケーリングを制限しているサイトは、モバイルで19%・デスクトップで21%のみです。これは2024年比で1〜2%の改善であり、ゆっくりではありますが着実な進歩を示しています。

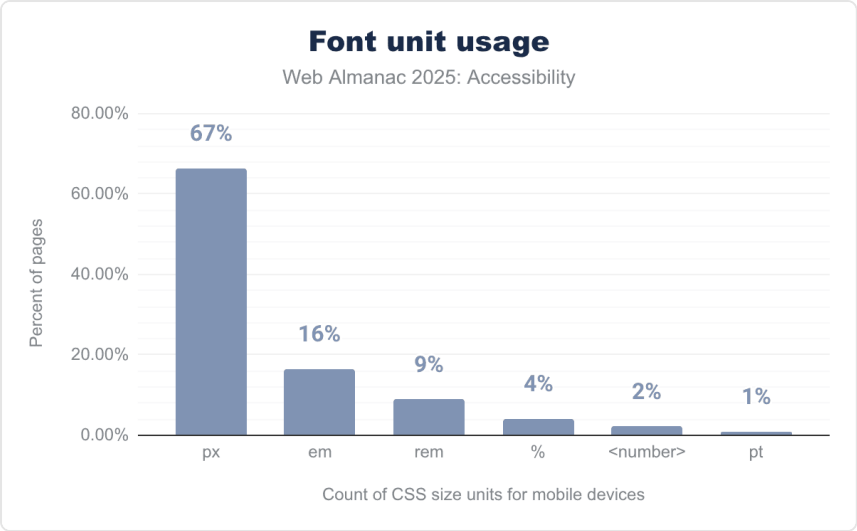


図6.5. フォントサイズ単位の使用状況。

フォントサイズの単位は、テキストがユーザー設定にどのように反応できるかに直接影響します。`em` や `rem` などの相対単位を使うと、ブラウザの設定に合わせてテキストが予測可能にスケーリングされます。2025年には、モバイルサイトでの `em` の使用が2%増加し、読みやすさのためにフォントサイズを調整するユーザーの体験が改善しました。それ以外のフォントサイズ単位の使用状況は昨年とほぼ同じです。

サイトがズームを制限しているかどうかを確認するには、ソースコードで `<meta name="viewport">` タグを調べてください。リサイズを制限する `maximum-scale`・`minimum-scale`・`user-scalable=no`・`user-scalable=0` などの値の使用を避けてください。代わりに、ユーザーがコンテンツサイズを自由に調整できるようにしましょう。WCAGはコンテンツや機能の損失なしにテキストが200%までリサイズできることを要求しています[165]。

言語の識別

`lang` 属性でページの主要言語を宣言することは必須です。スクリーンリーダーが正しい発音規則を選択でき、ブラウザがより正確な自動翻訳を提供できるようになります。主要言語を超えて、メイン言語と異なるセクションの言語を指定することも同様に重要です。これにより、スクリーンリーダーが外国語の単語やフレーズに対して発音を適切に切り替えられるようになります。

165. https://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-scale.html#:~:text=Content%20satisfies%20the%20Success%20Criterion%20if%20it%20can,more%20extreme%2C%20adaptive%20layouts%20may%20introduce%20usability%20problems.

straightforward Level A WCAG requirement[166]であるにもかかわらず、言語宣言は多くのサイトが不十分な領域のままです。2025年には、約86%のサイトが有効な `lang` 属性を含んでおり、2024年からほぼ変わっていません。これは継続的な採用を示す一方、改善の余地があることも浮き彫りにしています。

`lang` 属性の正しい適用は、ページの主要言語を指定するために `<html>` タグに含めることから始まります。ページには複数の言語が含まれることがよくあります。必要に応じて個々の要素やセクションに `lang` 属性を使用してください。W3Cの言語の部分指定に関するドキュメント[167]にこのトピックの詳細なガイダンスがあります。

言語宣言の欠如や誤りは翻訳エラーを引き起こす可能性があります。

例えば、Chromeの自動翻訳は宣言された言語がない場合にページコンテンツを誤解釈し、混乱した不正確な翻訳につながる可能性があります。適切な言語タグ付けは、右から左に書く言語のスタイリングや他の言語固有の動作もサポートします。

ユーザー設定

モダンCSSには、ウェブサイトがユーザーのオペレーティングシステムまたはブラウザ設定に適応できるユーザー設定メディアクエリ[168]が含まれています。ユーザーはより快適でパーソナライズされた体験を得られます。ウェブサイトはモーション・コントラスト・カラースキームの設定に対応できます。

166. *https://www.w3.org/WAI/WCAG22/Understanding/language-of-page*
167. *https://www.w3.org/WAI/WCAG21/Understanding/language-of-parts.html*
168. *https://developer.mozilla.org/docs/Web/CSS/Reference/At-rules/@media/prefers-color-scheme*

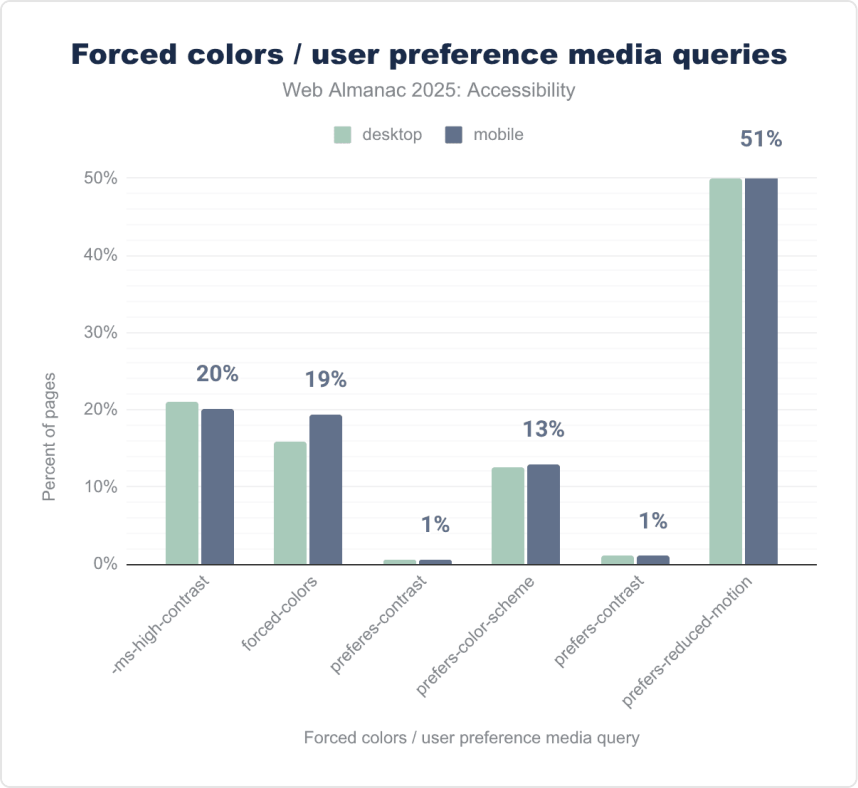


図6.6. ユーザー設定メディアクエリ。

最も馴染み深いクエリである `prefers-reduced-motion` と `prefers-color-scheme` はブラウザで広くサポートされ続けています。2025年には、これらのクエリのウェブサイトによる採用はほとんど変化がありません。しかし、低視力ユーザーのハイコントラストモードをサポートする `forced-colors` の使用は5%増加して19%になりました。一方、時代遅れの `-ms-high-contrast` メディアクエリの使用は3%減少して20%になりました。これはモダンCSSスタンダードへの段階的な移行を反映しています。

これらの設定を継続的に取り入れることで、個人のニーズとシステム設定を尊重し、アクセシビリティとユーザー満足度が向上します。

CSSメディアクエリによるパーソナライズのより広い実装は、これらの段階的な改善にもかかわらず大きな成長は見られていません。さらなる採用を促進することで、前庭障害の敏感さに対するモーション低減や視覚的な快適さのためのディスプレイカラーやコントラストの調整など、ウェブサイトがユーザーの設定を尊重できるようになります。

ナビゲーション

ユーザーはさまざまな方法でウェブサイトをナビゲートします。マウスでスクロールする人もいれば、キーボード・スイッチコントロールデバイス・スクリーンリーダーで見出しをナビゲートする人もいます。効果的なナビゲーションシステムは、入力デバイスに関わらずすべてのユーザーに機能しなければなりません。

ワイドスクリーンテレビやSiri・Amazon Alexaのような音声インターフェースは独自のナビゲーション課題を生み出します。サイトに適切なセマンティック構造を構築することで、スクリーンリーダーユーザーのナビゲーションが向上します。また、他の多くの種類のテクノロジーのユーザーにも役立ちます。

フォーカス表示

フォーカス表示は、ウェブコンテンツを移動するためにキーボードナビゲーションと支援デバイスに主に依存するユーザーにとって必須です。現在フォーカスされている要素を強調表示する視覚的な手がかりを提供し、ユーザーがページ上のどこにいるかを理解できるようにします。

Google Lighthouseなどの自動テストツールは多くの基本要件を特定し、フォーカスインジケーターに関する明らかな問題を検出できます。しかし、キーボードトラップ・フォーカス順序・フォーカスが新しいコンテンツに論理的に移動するかどうかなどの複雑なインタラクションには限界があります。自動監査に合格することは、サイトのキーボードアクセシビリティやキーボードユーザーの良好なユーザー体験を保証しません。

包括的な手動テストは欠かせません。

オープンソースのAccessibility Insights for the Web[169]拡張機能などのツールは、Dequeのaxeコアとガイドとなる AST を組み合わせて、ガイド付きの手動テストを提供しています。「Tab Stops」の可視化機能は、テスターがキーボードユーザーが辿るパスを確認し、フォーカススタイルの欠如や予期しないフォーカストラップなどの潜在的な問題を効果的に特定するのに役立ちます。

運動能力に制限のある代替ナビゲーションデバイスのユーザーは、フォーカスの視認性と順序に関して固有のニーズを持っています。支援技術のインターフェースをカスタマイズすることで、能力に合わせたコントロールを最大化できます。

フォーカステストのベストプラクティスには以下が含まれます：

169. *https://accessibilityinsights.io/docs/web/overview/*

- キーボードユーザーが行き詰まるフォーカストラップがないこと[170]
- すべてのインタラクティブコントロールがキーボードでフォーカス可能であること[171]
- タブ順序が論理的で直感的であること[172]
- フォーカスが新しくまたは動的にロードされたコンテンツに適切に向けられること[173]

A11y Collectiveのフォーカスインジケーターに関するレポート[174]は、可視フォーカスアウトラインスタイルの実装とテストのための実践的な洞察を提供しています。

フォーカススタイル

WCAGはすべてのインタラクティブコンテンツに明確に見えるフォーカスインジケーター[175]が必要であることを定めています。この視覚的な手がかりにより、キーボードユーザーはページを移動しながら現在フォーカスされている要素を識別できます。

目立つフォーカスインジケーターがないと、キーボードユーザーや支援技術に依存するユーザーは迷子になりやすくなります。ハイコントラストアウトラインなどの堅牢なフォーカススタイルは、アクセシブルなデザインの基本です。GOV.UK[176]のような多くの機関では、一貫性と明確性を確保するためにフォーカスインジケーターの標準を確立しています。

デザインアノテーションはCraig AbbottがTetraLogicalブログで明確に示したように[177]、キーボードインタラクションを指定する必要があります。この投稿の直後、GitHubは同じ問題に対処するためにアクセシビリティ Annotation Toolkit[178]をリリースしました。

170. *https://developer.chrome.com/docs/lighthouse/accessibility/focus-traps*
171. *https://developer.chrome.com/docs/lighthouse/accessibility/focusable-controls*
172. *https://developer.chrome.com/docs/lighthouse/accessibility/logical-tab-order*
173. *https://developer.chrome.com/docs/lighthouse/accessibility/managed-focus*
174. *https://www.a11y-collective.com/blog/focus-indicator/*
175. *https://www.w3.org/WAI/WCAG21/Understanding/focus-visible.html*
176. *https://www.gov.uk/*
177. *https://tetralogical.com/blog/2025/09/23/annotating-designs-using-common-language/*
178. *https://github.com/github/annotation-toolkit*

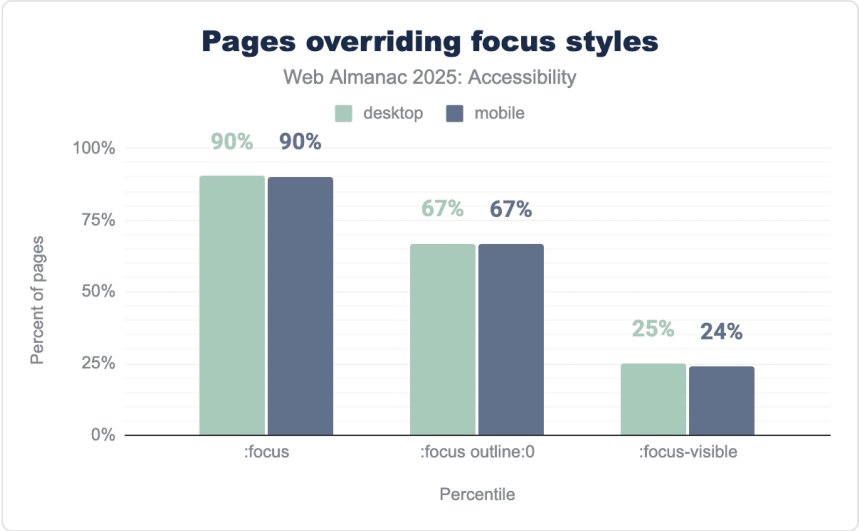


図6.7. ブラウザのフォーカススタイルを上書きしているページ。

2025年には、67%のサイトがデフォルトのフォーカスアウトラインを明示的に削除しており、2024年から14%増加しました。この懸念すべきトレンドは、効果的なスタイルに置き換えられない場合、アクセシビリティを損なう可能性があります。一方、`:focus-visible`疑似クラスの採用は増加しています。これは、開発者が必要な時にのみ表示されるコンテキスト対応のフォーカスインジケーターを作り始めていることを意味します。

`tabindex`

`tabindex`属性は、要素のキーボードフォーカス順序への参加を制御します。開発者はフォーカス可能な要素を含める・除外する・並び替えることができます。正しく使用することは論理的なナビゲーションとアクセシビリティをサポートし、WCAGの要件[179]でもあります。特に正の値での誤用は、自然なタブ順序を乱し、ユーザーを混乱させる可能性があります。

179. https://www.w3.org/WAI/WCAG21/Understanding/focus-order.html

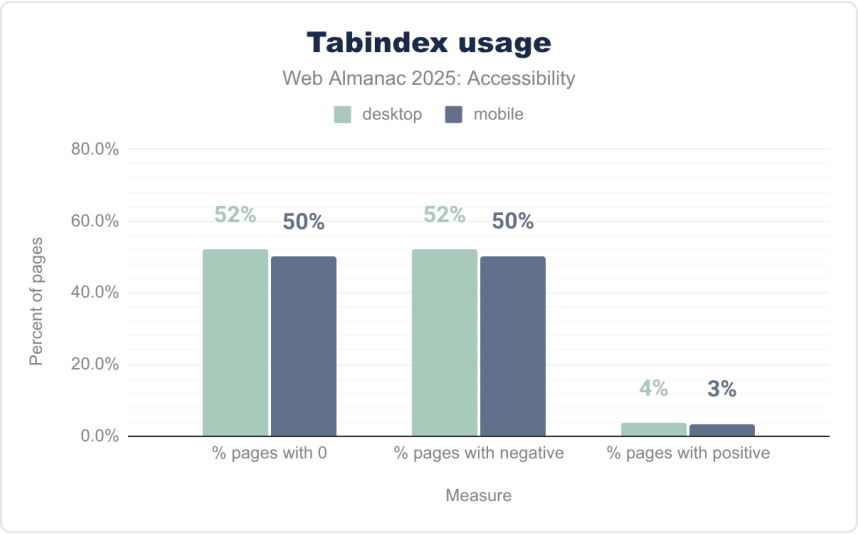


図6.8. `tabindex` の使用状況。

2025年には `tabindex` の使用がわずかに増加しました。50%超のサイトが使用しており、2024年より約3〜4%高くなっています。正の `tabindex` の使用は安定して低い水準を保っており、正の tabindex 値を避けるべきという認識が続いていることを反映しています。

ランドマーク

ランドマークは `<header>` ・ `<nav>` ・ `<main>` ・ `<footer>` などのネイティブHTML要素を使用して、ウェブページを明確なテーマ別領域に構造化します。これらの要素は明確で高レベルなページのアウトラインを作成し、支援技術のユーザーがレイアウトをすばやく理解して関連セクションに直接ジャンプできるようにします。

開発者が冗長なARIA属性を追加する場合、一般的なアクセシビリティのアンチパターンが見られます。例えば、 `<nav>` 要素に `role="navigation"` を追加するケースです。 `<nav>` 要素は本質的にナビゲーションロールを持っているため、この重複はコードを煩雑にするだけで利益なく、支援技術を混乱させる可能性があります。ARIAランドマークロールを追加する前に、まずネイティブHTML5要素を優先するのがベストプラクティスです[180]。これはARIAの主要なガイドラインです。

Eric Baileyのようなアクセシビリティの専門家は、ネイティブのセマンティックHTMLで十分なコンテキストでARIAを過度に使用することの落とし穴[181]を強調しています。Heydon

180. https://www.w3.org/WAI/WCAG21/Techniques/html/H101
181. https://www.smashingmagazine.com/2025/06/what-i-wish-someone-told-me-aria/

Pickeringのウェブアクセシビリティの12原則[182]も、セマンティック構造とランドマークがアクセシブルなナビゲーションで果たす重要な役割を強調しています。

要素	要素 %	ロール %	両方 %
main	40.72%	17.81%	47.34%
header	65.99%	10.95%	67.41%
nav	67.73%	18.02%	70.94%
footer	66.38%	9.59%	67.66%

図6.9. ランドマーク要素とロールの使用状況（デスクトップ）。

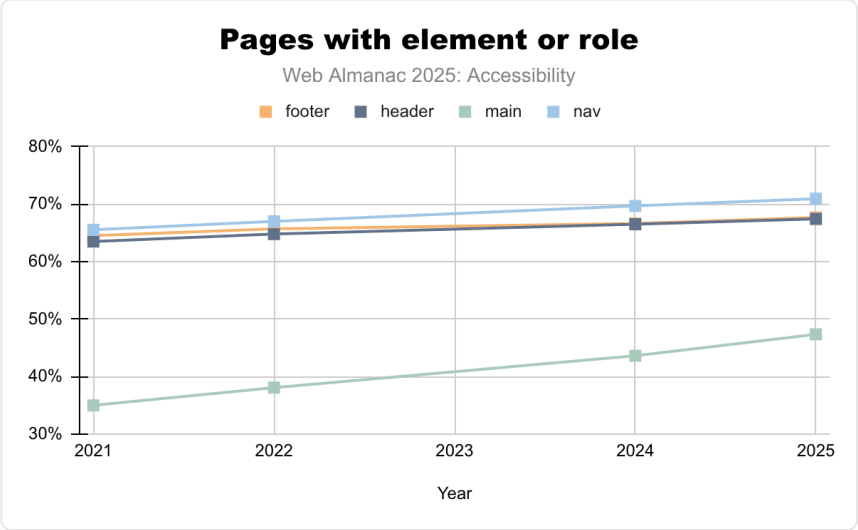


図6.10. 要素ロールを持つページの年次成長。

2025年には、ネイティブの `<main>` 要素の使用拡大（現在47%、2024年比3%増）を筆頭に、ARIAランドマークの採用がわずかに増加しました。この進歩は、セマンティックHTMLへの準拠の改善と、支援ツールに依存するユーザーのためのより堅牢なページ構造を反映しています。

スクリーンリーダーユーザーは、これらのページ領域間をジャンプするために「ローター」やランドマークメニューを使ってナビゲートすることが多いです。ランドマークを指すスキップリンクは、コアコンテンツへの即時アクセスを可能にすることでユーザビリティを向上させます。繰り返されるナビゲーションブロックやバナーを回避できます。スキップリンク

182. *https://heydonworks.com/article/pride-shame-and-accessibility/*

については後のセクションで説明します。

ネイティブHTML5ランドマークの活用と冗長なARIAロールの最小化に関する継続的な教育により、キーボードと支援技術のナビゲーション体験がさらに向上します。セマンティック構造の採用の増加はアクセシビリティの目標をサポートし、ウェブコンテンツを現代のベストプラクティスに合わせます。

見出し階層

一貫した見出し構造はウェブページの目次のように機能します。アクセシビリティ・SEO・ユーザーの理解をサポートします。スクリーンリーダーユーザーにとって、見出しによるナビゲーションは情報をすばやく見つけるための重要な方法です。検索エンジンもページの構成と関連性を理解するために見出し階層に依存しています。

見出しは視覚的なスタイリングだけでなく、ドキュメント構造を伝えるべきです。`<h3>` や `<h4>` などの見出しタグをフォントサイズのためだけに使用すると、論理的な順序が崩れます。支援技術のユーザーのナビゲーションが困難になり、構造とプレゼンテーションの分離原則に違反します。代わりに、開発者はCSSで見出しをスタイリングし、コンテンツ階層に従って見出しタグを使用すべきです。

セマンティクスが重要な理由の復習として、Jono Aldersonのこの記事[183]を参照してください。

複数年にわたる低下の後、見出し階層のスコアは2025年に約2%改善し、適切な見出し構造への関心の再燃を示しています。

図6.11. 適切に順序付けされた見出しのLighthouse監査に合格しているモバイルサイト。

それでも、構造ではなくスタイリングのために見出しを誤用することは一般的なままです。

スキップリンク

スキップリンクは、キーボードユーザーやマウス以外の入力デバイスを使用するユーザーが、サイトのナビゲーションメニューなどの大きくて繰り返しのあるコンテンツブロックをスキップして、メインページコンテンツに直接ジャンプできるようにするナビゲーション補助機能です。通常、効率的なナビゲーションのために「メインコンテンツにスキップ」リン

183. *https://www.jonoalderson.com/conjecture/why-semantic-html-still-matters/*

クがページの最初のフォーカス可能な要素として存在します。

コンテンツのブロックをバイパスすることはWCAGの要件[184]であり、基本的な実装が標準のままです。

PayPalのオープンソースSkipTo[185]のような高度なツールは、ページ上のすべての主要なランドマークと見出しの動的メニューを生成するために存在しています。このより豊かなインタラクションは幅広いユーザーに利益をもたらし、全体的なナビゲビリティとユーザビリティを向上させます。Eleanor Hecksはキーボードアクセシビリティの重要性[186]について説得力のある記事を書き、TetraLogical[187]も同様の記事を書きました。

スキップリンクの採用は2024年から2025年にかけてほぼ静的なままです。

図6.12. スキップリンクを持つと思われるモバイルおよびデスクトップページ。

デスクトップとモバイルページの約24%が、一般的な分析方法で検出可能なスキップリンクを含んでいます。この数値は実際の使用状況を過小評価している可能性があり、一部のスキップリンクはページのより深い部分に出現したり、ナビゲーションメニュー以外のランドマークをターゲットにしている場合があるためです。

184. *https://www.w3.org/WAI/WCAG21/Understanding/bypass-blocks.html*
185. *https://paypal.github.io/skipto/*
186. *https://www.smashingmagazine.com/2025/04/what-mean-site-be-keyboard-navigable/*
187. *https://tetralogical.com/blog/2025/05/09/foundations-keyboard-accessibility/*

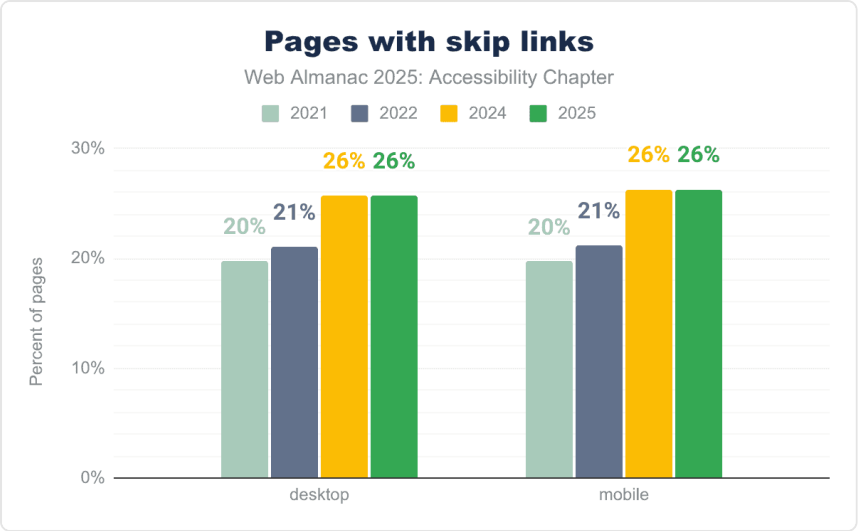


図6.13. スキップリンクを持つページの年次成長。

ドキュメントタイトル

説明的なページ `<title>` は基本的な必要条件です。ブラウザのタブやウィンドウ間をナビゲートするユーザーにコンテキストを提供し、スクリーンリーダーが最初にアナウンスする情報であることが多く、ユーザーが状況を把握するのを助けます。WCAGもすべてのページに意味のあるタイトルが必要[188]であることを定めています。

188. https://www.w3.org/WAI/WCAG21/Understanding/page-titled

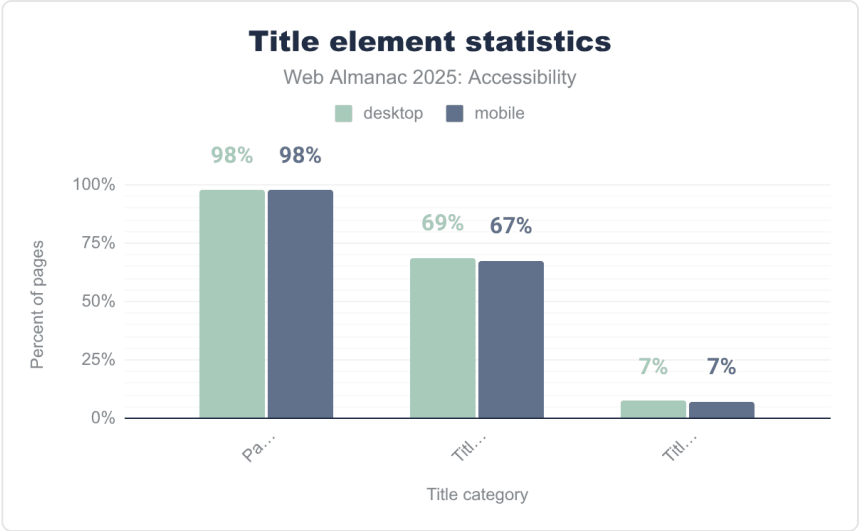


図6.14. titleタグ要素の統計。

2025年のデータは、前年と比較してドキュメントタイトルの存在と説明性がわずかに改善されたことを示しています。現在、約98%のサイトが `<title>` 要素を含んでおり、2024年から1%増加しています。

これは肯定的です。しかし、この高い包含率にもかかわらず、多くのタイトルの説明性は不十分なままです。これはユーザビリティに影響し、特に方向確認のために明確なタイトルに依存するスクリーンリーダーユーザーに影響します。

4語以上のタイトルを持つモバイルサイトが2%減少しており、モバイルページでより短いまたは具体性の低いタイトルが増えていることを示している可能性があります。ナビゲーションとコンテキストを強化するために、ページコンテンツの簡単な説明とウェブサイト名の両方を含めることが引き続きベストプラクティスです。

ドキュメントタイトルはすべてのユーザーに利益をもたらす基本的なアクセシビリティ機能のままです。ブラウザのタブやウィンドウをナビゲートする際のコンテキストを提供します。ほぼすべてのページに存在していますが、タイトルの説明性と一貫性の改善は2025年以降も重要な焦点であり続けます。

テーブル

HTMLテーブルは2次元のグリッドでデータを表示します。アクセシビリティは適切なセマンティック要素での構造化に依存します。`<caption>` を使用するとスクリーンリーダーユーザーに重要なコンテキストが提供され、`<th>` 要素は行と列のヘッダーを定義してユーザ

ーがデータ内の関係を理解するのを助けます。2025年にリリースされたSteve Faulknerのツール[189]は、開発者がHTML要素のセマンティクスを迅速に検査するのに役立ちます。

`<caption>` の使用は2024年と比較して2025年も安定しており、キャプションを含むサイトの割合はわずかです。この低い採用率は過去年と同様です：デスクトップサイトの約1.6%がキャプションを含んでおり、これは重要ではあるものの見落とされがちなアクセシビリティ機能です。

テーブルをレイアウト目的で誤用するべきではありません。CSS FlexboxとGridがレイアウトを担います。テーブルが純粋にレイアウト用途で使用される場合、`role="presentation"` 属性でセマンティックな意味を除去し、支援技術との混乱を避けます。

	desktop	mobile	desktop	mobile
キャプション付きテーブル	5.6%	4.7%	1.7%	1.7%
プレゼンテーション用テーブル	4.9%	5.4%	3.6%	4.8%

図6.15. テーブルの使用状況。

2025年には、モバイルテーブルの4.9%がこの技術を使用しており、2024年の4%、2022年の1%から増加しています。テーブルをアクセシブルにするためにセマンティックHTML要素を正しく使用することへの重点は変わっていません。

フォーム

フォームはログインから購入、情報共有まで、ユーザーがウェブと対話する方法です。フォームのアクセシビリティを確保することは、ユーザーがタスクを完了しオンラインで十分に参加するために重要です。

`<label>` 要素

`<label>` 要素は入力フィールドにアクセシブルな名前を提供するための標準的な推奨方法であり続けています。通常は入力の `id` を指す `for` 属性を通じて説明的なテキストをフォームコントロールにプログラム的に関連付けることで、支援技術のユーザーが必要な情報を明確に理解できるようにします。適切なラベルは、ラベルをクリックすると入力にフォーカスが設定されるため、クリック可能な領域を拡大することでユーザビリティも向上させます。

189. *https://codepen.io/stevef/pen/ByoMebv*

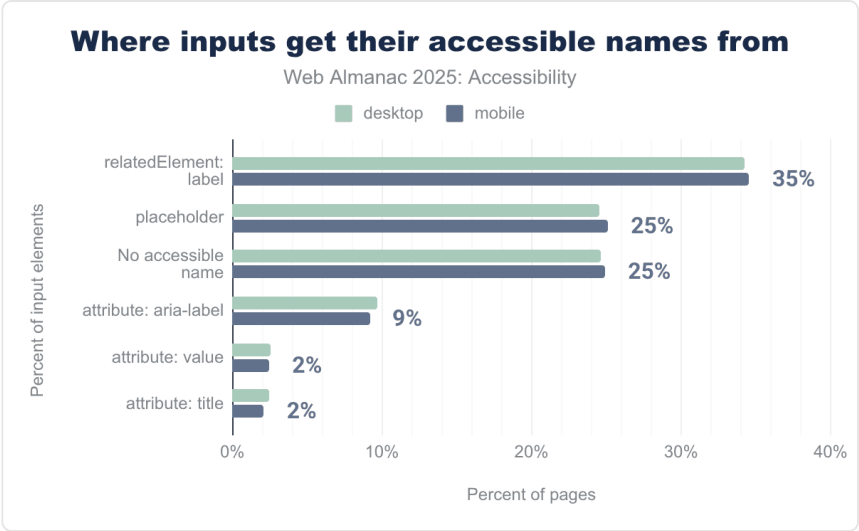


図6.16. 入力のアクセシブルな名前の取得元。

2025年には、モバイル入力の約35%が `<label>` からアクセシブルな名前を取得しており、2024年の32%から増加しています。これは肯定的なトレンドです。

プレースホルダーテキストのみからアクセシブルな名前を取得する入力が2%減少しました。プレースホルダーテキストの信頼性は低く、ラベルの代替にはなりません。しかし、アクセシブルな名前がまったくない入力の割合は昨年から変わっておらず、継続的なアクセシビリティのギャップを示しています。

2025年のデータは `label` の使用の段階的な改善を示しています。また、完全なアクセシビリティ準拠とユーザビリティを達成するために、適切なラベリング慣行を引き続き拡大する必要性も強調しています。

`placeholder` 属性

placeholder属性はフォームフィールド内に期待される入力形式のヒントや例を提供します。しかし、その入力のアクセシブルな名前として `<label>` 要素を置き換えるべきではありません。プレースホルダーテキストはユーザーが入力を開始するとすぐに消え、参照できなくなります。

プレースホルダーテキストはデフォルトのコントラストが低く、WCAGのカラーコントラスト要件に失敗することがよくあります。スクリーンリーダーのプレースホルダーサポートも広く異なります。

推奨されるアプローチは、入力に対して可視で、プログラム的に関連付けられたラベルを使用し、プレースホルダーは補足的なヒントや例としてのみ使用することです。

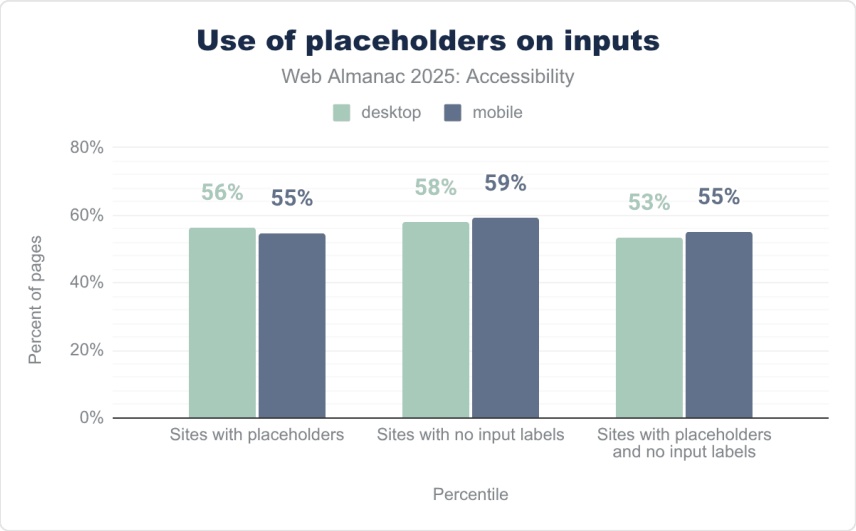


図6.17. 入力でのプレースホルダーの使用。

2025年には、プレースホルダーテキストのみを入力のアクセシブルな名前として使用するケースが2%減少しました。この肯定的なトレンドにもかかわらず、この慣行は依然として一般的です。デスクトップ入力の53%・モバイル入力の55%がアクセシブルな命名のためにプレースホルダーテキストのみに依存しており、依然として重大なアクセシビリティの障壁をもたらしています。

必須情報

フォームフィールドが必須であることを伝えることは、ユーザビリティとアクセシビリティにとって重要です。アスタリスク（*）などの視覚的なインジケーターは一般的ですが、セマンティック情報が欠けているため単独では不十分です。

HTML5の `required` 属性は、フォームを送信する前にユーザーがフィールドに記入しなければならないことを示すネイティブで機械可読な方法を提供します。この属性はテキスト・メール・パスワード・日付・チェックボックス・ラジオなど多くの入力タイプで機能します。ブラウザはバリデーションを強制し、支援技術はユーザーに必須ステータスを伝えます。

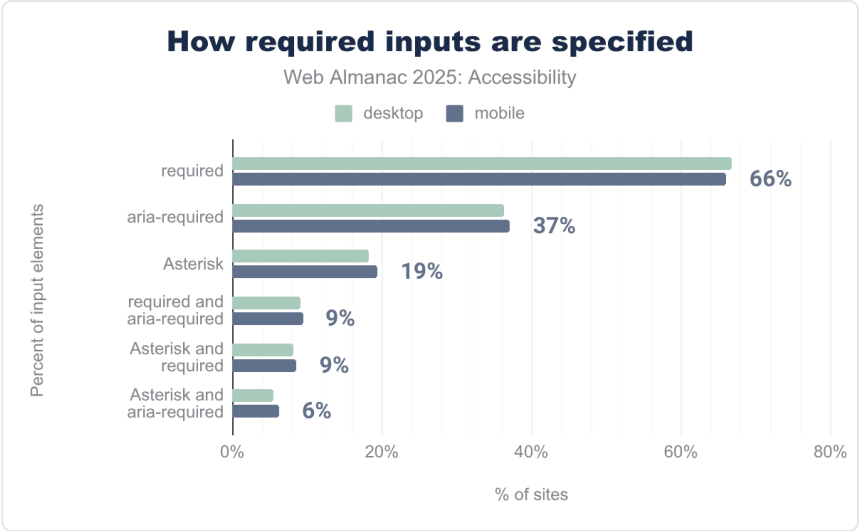


図6.18. 必須入力の指定方法。

2025年には required属性の採用がわずかに増加し、モバイルで1%増の66%になりました。`aria-required` の使用はモバイルで3%減少して37%になりました。これは、ARIAよりもネイティブHTMLバリデーションのより意味論的な使用への段階的なシフトを示しています。ARIAはネイティブセマンティクスを置き換えるためではなく、補完するためのものです。

2025年の進歩は、必須入力のより良いセマンティック表示に向けたゆっくりではありますが着実な動きを反映しており、フォームのアクセシビリティとユーザー体験を改善しています。

キャプチャ

CAPTCHAは人間と自動ボットを区別し、悪意のある活動を軽減します。頭字語は「Completely Automated Public Turing Test to Tell Computers and Humans Apart（コンピュータと人間を区別するための完全自動化公開チューリングテスト）」を意味します。CAPTCHAは必要なセキュリティ機能を果たしますが、特に視覚・運動・認知の障害を持つ人々にとって重大なアクセシビリティの障壁を頻繁に生み出します。

従来の視覚的なCAPTCHAは、障害を持つユーザーにとって解決が困難または不可能な場合があります。W3Cはより包括的な代替検証方法の検討を推奨しており、例えば：

- 音声による挑戦を提供するオーディオCAPTCHA

- ユーザー入力なしにバックグラウンドで動作する「invisible」CAPTCHA
- 多要素認証方法やよりシンプルな検証フローの組み込み。

2025年には、CAPTCHAの使用は前年と比較してほぼ安定したままです。

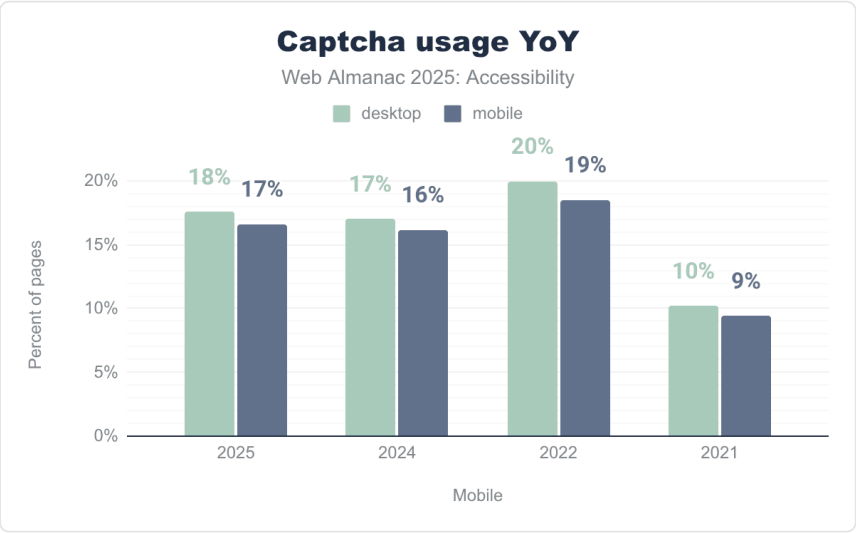


図6.19. CAPTCHAの使用状況（前年比）。

特筆すべきは、ルクセンブルク政府がCAPTCHAスキャナー[190]ツールをリリースしたことです。これは政府ウェブサイト全体のCAPTCHA実装を評価・監視し、公共部門のアクセシビリティ準拠の改善を目的としています。

視覚的なCAPTCHAをよりアクセシブルなオプションに置き換えるまたは補完する継続的な努力は不可欠です。すべてのユーザーが過度の困難や排除なしに検証ステップを完了できるべきです。

ウェブ上のメディア

ウェブ上のアクセシブルなメディアには、コンテンツをすべての人が使用できるようにするための代替フォーマットの提供が必要です。視覚障害のあるユーザーは、重要な視覚情報を伝える音声説明から利益を得ます。聴覚障害のあるまたは難聴のユーザーは、音声コンテンツにアクセスするためにキャプションや手話通訳に依存しています。

190. *https://accessibilite.public.lu/en/news/2025-09-22-captchas.html*

音声説明とキャプションだけでは十分ではありません。音声のみおよびビデオのみのコンテンツには、完全なテキスト代替を提供するトランスクリプトが必要です。画像などの非テキストコンテンツには適切な代替テキストを提供してください。意味のある情報を追加しない場合は、装飾的なものとしてマークしてください。

アクセシブルなメディアの原則と要件は2024年から2025年にかけて一貫しており、障害のあるユーザーにインクルーシブなマルチメディア体験を提供することの継続的な重要性を強調しています。

画像

`alt` 属性は画像のテキスト説明を提供します。スクリーンリーダーユーザーが視覚コンテンツを理解するために必須です。2025年でも、この属性は画像アクセシビリティの基本として残っており、前年からエラー率に大きな変化はありません。

図6.20. `alt` テキストを持つ画像のLighthouse監査に合格している画像の割合。

JPGとPNGファイルはウェブ画像を引き続き支配していますが、WEBPとSVG形式の使用には歓迎すべき成長が見られます。SVGファイルは複雑でインタラクティブな画像に役立つ豊かなセマンティクスを提供します。

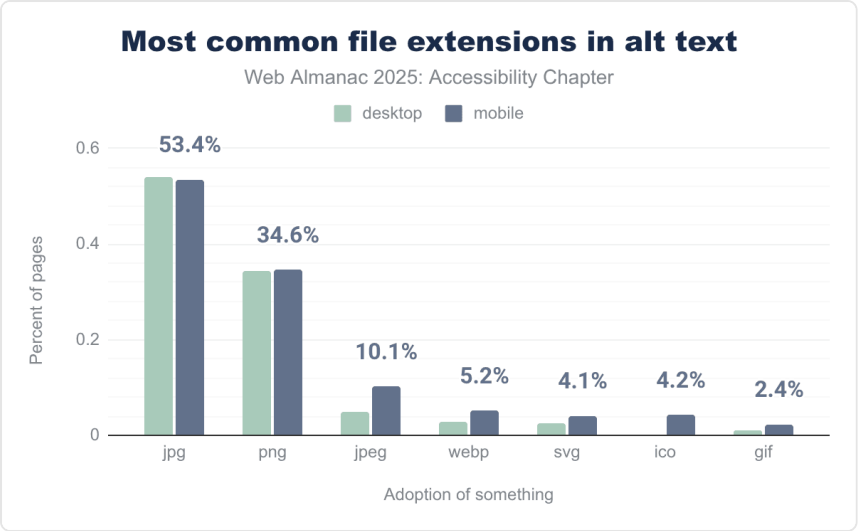


図6.21. `alt` テキストで最も一般的なファイル拡張子。

しかし、引き続き継続している問題が1つあります：画像のalt textの約8.5%が `.jpg` や `.png` などの一般的なファイル拡張子で終わっています。これは自動オーサリングツールがファイル名を `alt` テキストとして挿入する場合によく起こります。残念ながら、これは何の価値も追加せず、支援技術に依存するユーザーの助けになりません。

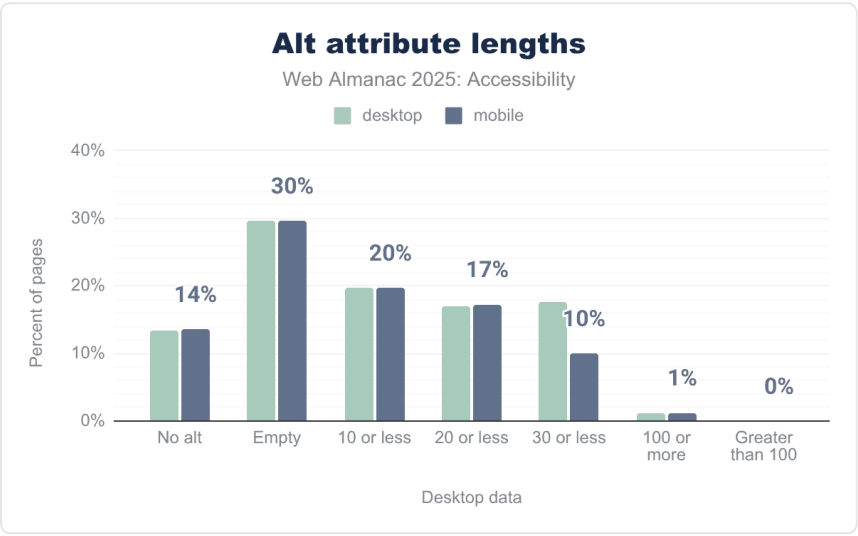


図6.22. `alt` 属性の長さ。

説明性と簡潔さのバランスが取れる傾向にある20〜30文字のalt textへのポジティブなトレンドがあります。しかし、約50%の画像はまだ空のalt属性か10文字未満のテキストを持っています。空のalt textは純粋に装飾的な画像にのみ適切です。ただし、ほとんどの画像は意味のある説明に値する重要な情報を伝えており、意味のあるalt textから恩恵を受けます。

ベストプラクティスは引き続き、画像コンテキストに合わせた簡潔かつ説明的なalt textの提供、ファイル名の回避、適切な場合のSVGのようなセマンティックファイル形式の使用を強調しています。人工知能（AI）ツールも有望です。DrupalのAI支援alt textサジェスト機能の統合[191]は、著者が編集可能な例を提供することでより良いalt属性を作成するのを助けます。Brian TeemanはAI生成のAlt Text[192]について興味深い批評を書いています。

画像は着実な進歩にもかかわらず、大幅なアクセシビリティ改善の機会がある分野であり続けています。

音声と動画

HTML `<track>` 要素は、 `<video>` や `<audio>` などのメディア要素のキャプション、字幕、音声説明などの時間指定テキストトラックを提供します。2025年においても、この要素はまだ十分に活用されていません。聴覚障害、難聴、または視覚障害のあるユーザーにとっての重要性にもかかわらず、採用率は1%未満と非常に低いままです。

多くの現代的なビデオプラットフォームは現在、ネイティブの `<track>` 要素の代わりにHTTP Live Streaming (HLS)[193]を一般的に使用しています。これが低い使用統計の一因となっているかもしれません。カスタム実装が必要なHLSは、開発者が自分でキャプションサポートを組み込むことをより意識的に行う必要があることを意味します。つまり、開発者がキャプションを忘れたり、不十分な実装をしたりする余地がより多くあります。

キャプションは聴覚障害や難聴のあるユーザーに不可欠です。また、騒がしい環境にいる視聴者や音声言語の理解が困難な方にも利益をもたらします。音声説明は視覚障害のあるユーザーが視覚的コンテンツについてのコンテキストを得るのを可能にします。

2024年と比較して、キャプションと字幕の `<track>` 使用に大きな成長は見られておらず、デスクトップで0.1%から0.4%、モバイルで0.1%から0.2%への微増にとどまっています。これは業界にまだ大幅な改善の余地があることを示しています。

ARIAを使った支援技術

Accessible Rich Internet Applications（ARIA）は、ウェブコンテンツのアクセシビリティを

191. *https://www.drupal.org/project/ai_image_alt_text*
192. *https://magazine.joomla.org/all-issues/june-2024/ai-generated-alt-text*
193. *https://en.wikipedia.org/wiki/HTTP_Live_Streaming*

向上させるためのHTML属性のセットです。ARIAは、ネイティブHTMLだけではアクセシブルにできない複雑またはカスタムコンポーネントに特に有用です。ARIAは動的でインタラクティブなユーザーインターフェイスを強化し、スクリーンリーダーやその他の支援技術を使用する人々がすべてのページ要素を理解してインタラクトできるようにします。

ARIAは慎重に使用する必要があります。

誤った使用や過度な使用は新たな障壁を生む可能性があり、ユーザーとアクセシビリティツールの両方に混乱を引き起こします。例えば、意図された機能と一致しないARIA属性、不適切な要素に追加されたロール、または冗長なARIAはユーザー体験を乱し、アクセシビリティエラーを増やす可能性があります。

Adrian Roselliの研究は、`aria-description`などの特定のARIAプロパティの限界[194]を強調し、ARIAの強みと落とし穴の両方を理解することの重要性を示しています。

ARIAの最も重要な原則は次のとおりです:

ネイティブHTMLを使用できる場合は、そうするべきです。

`<button>`、`<input>`、`<nav>`などのネイティブ要素には、ARIAが完全には再現できない組み込みのアクセシビリティがあります。ARIAは必要な場合にのみネイティブセマンティクスを補完すべきであり、置き換えるものではありません。

Florian Schroiffを含む専門家による最近のガイダンス[195]と現在のベストプラクティスは、複雑なカスタム要素にのみARIAを適用し、偶発的な排除や誤通信を避けるために仕様に厳格に従うことを強調しています。

2025年においても、ARIAはウェブアクセシビリティにおいて重要ですが、時として問題のある役割を果たし続けています。

ARIAロール

ARIAロールは要素の目的や種類を支援技術に伝えます。2025年においても、ウェブコンテンツをアクセシブルにする上で重要な役割を果たし続けています。`<button>`などのネイティブHTML要素には組み込みのセマンティクスがありますが、ARIAはタブインターフェイスや`dialog`コンポーネントなど、ネイティブな等価物がないカスタムコンポーネントにロールを割り当てる機能を提供します。

194. *https://adrianroselli.com/2025/01/aria-description-does-not-translate.html*
195. *https://www.a11y-collective.com/blog/aria-in-html/*

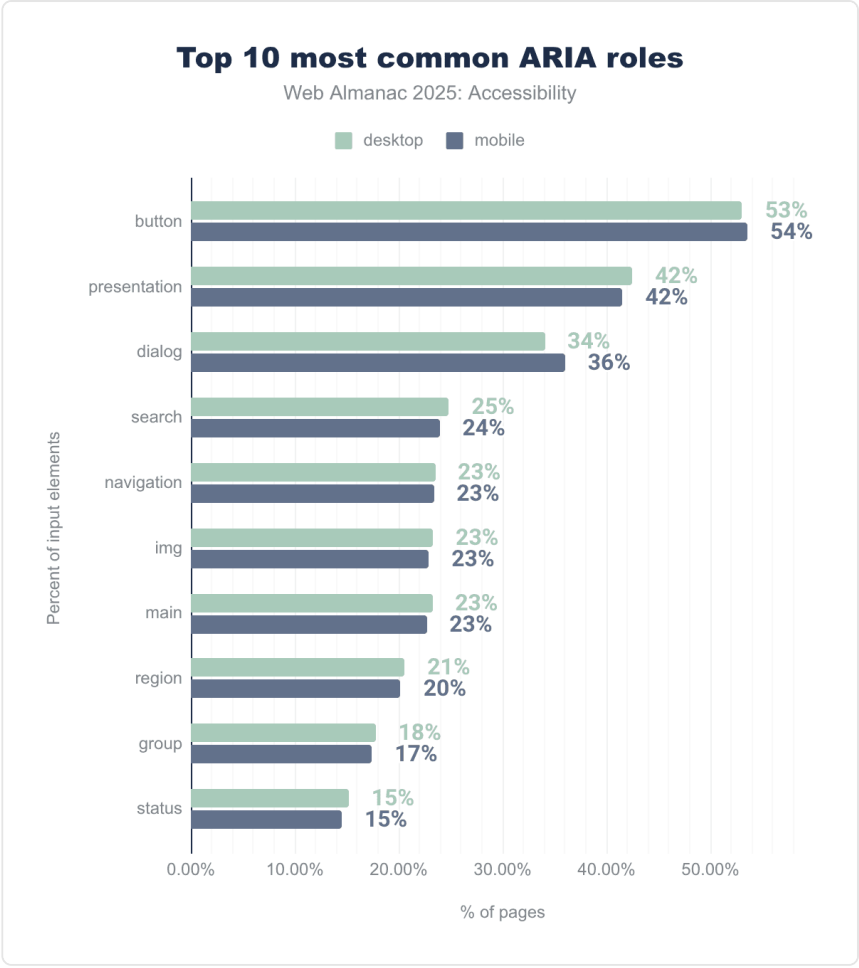


図6.23. 最も一般的なARIAロール トップ10。

2025年においてARIA `button` ロールの使用が約4%増加し、デスクトップで53%、モバイルサイトで54%に達しています。 `presentation` や `dialog` などのロールの使用も同様の増加が見られましたが、 `search` ロールの使用は安定しています。

ARIA `button` ロールの使用増加は懸念を引き起こします。これはウェブサイトが `<div>` や `<span>` などの非セマンティック要素に `button` などのロールを適用していることを示すことが多いです。あるいは、 `<button>` などのネイティブHTML要素に冗長にロールを割り当てているケースもあります。

presentationロールの使用

`role="presentation"` または `role="none"` を適用すると、支援技術に要素を純粋にプレゼンテーション的なものとして扱うよう指示します。これにより、アクセシビリティツリーからネイティブセマンティクスが削除されます。意味のある情報を伝えないレイアウト要素には有用ですが、過度な使用や誤用は重大なアクセシビリティの障壁を生む可能性があります。

例えば、`<ul>` 要素に `role="presentation"` を適用すると、子の `<li>` 要素のものを含む、リスト全体のセマンティクスが無視されます。スクリーンリーダーユーザーは、リストに何個のアイテムがあるかなどの重要なコンテキストと構造情報を失います。

`presentation` ロールは要素が純粋に装飾的またはレイアウト目的で使用される場合に誤解を招くセマンティクスを削除するのに役立ちますが、明確な意図を持って控えめに適用すべきです。

図6.24. デスクトップサイトとモバイルサイトの少なくとも1つの `presentation` ロールを持つサイトの割合。

2025年において、`role="presentation"` の使用が2%増加し、懸念すべきトレンドが続いています。

ARIAを使った要素のラベリング

ブラウザはアクセシブルな名前、ロール、状態、説明などのページ要素に関する情報を公開するアクセシビリティツリーを維持しています。支援技術はこれに依存してユーザーにコンテキストを伝えます。要素のアクセシブルな名前は重要で、通常は可視テキストコンテンツから導き出されます。ただし、ネイティブテキストが不十分または利用できない場合、`aria-label` や `aria-labelledby` などのARIA属性を使用してアクセシブルな名前を明示的に設定またはオーバーライドできます。

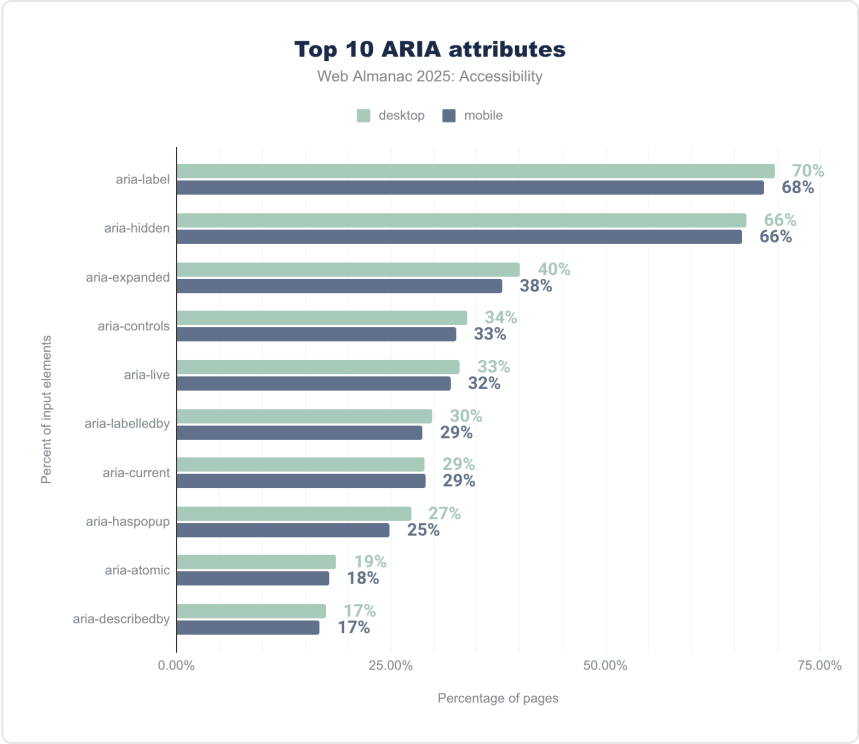


図6.25. ARIA属性 トップ10。

2025年において、主要なARIA属性のほぼすべての使用が増加しました。`aria-label` のデスクトップ使用は5%増加し、`aria-labelledby` は3%増加しました。デスクトップでの `aria-describedby` の使用は1%減少しました。

これらの変化は、開発者がよりプログラム的により多くの要素にアクセシブルな名前を割り当てていることを示しています。これは役立ちますが、慎重に実装されない場合には問題になる可能性もあります。

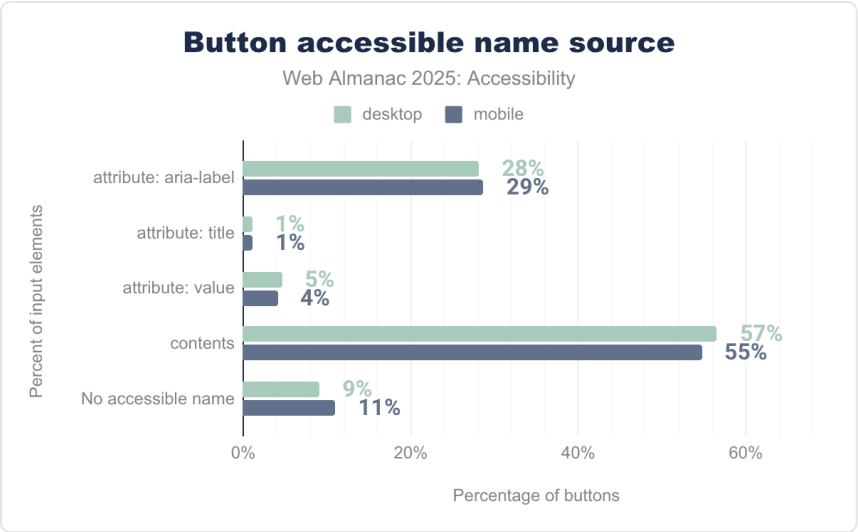


図6.26. ボタンのアクセシブルな名前のソース。

対応する可視ラベルなしに `aria-label` のみでボタンを定義することが4〜5%増加し続けているという懸念すべきトレンドが見られます。ユーザーが視覚的に見るものと支援技術が通知するものとのこの乖離は混乱と障壁を生む可能性があります。これは認知障害のある方や音声入力を使用する方に特に当てはまります。理想的には、一貫したユーザー体験を提供するためにアクセシブルな名前と可視ラベルが一致すべきです。

ページの約66%が `aria-label` 属性を使用しており、以前の年より増加し、アクセシブルな名前のための最も頻繁に使用されるARIA属性となっています。ページの約4分の1が `aria-labelledby` を使用しています。

アクセシビリティを向上させるためにARIAを使用することは肯定的ですが、意味のある正確な命名を確保するために支援技術を使ったテストと障害のあるユーザーの参加の重要性を強調しています。

コンテンツの非表示

`aria-hidden="true"` 属性は要素とそのすべての子孫をアクセシビリティツリーから削除し、スクリーンリーダーからコンテンツを見えなくします。これは、そうでなければ非視覚ユーザーを混乱させる可能性がある純粋に装飾的または冗長な視覚要素を隠すのに役立ちます。

図6.27. `aria-hidden` 属性を持つ少なくとも1つの要素があるページ。

2025年において、`aria-hidden` の使用は2024年と比較して3%増加しました。ウェブサイトの約66%がこのARIA属性を使用してコンテンツの一部を非表示にしています。

同様に、コンテンツのセクションが展開されているか折りたたまれているかを示す `aria-expanded` 属性も採用が増加し、デスクトップサイトの40%、モバイルサイトの38%に達しています。この属性はアコーディオンや展開可能なメニューなどの開示ウィジェットの状態を支援技術に伝えるために重要です。

これらのARIA属性の慎重な適用は2025年においても重要です。動的コンテンツの管理を助け、デバイスやユーザーニーズにわたってインクルーシブな体験を確保します。

スクリーンリーダー専用テキスト

2025年においても、アクセシビリティのための一般的かつ効果的な技術として、スクリーンリーダー専用テキストの使用があります。これは視覚的に非表示になっているが、スクリーンリーダーなどの支援技術からはアクセス可能なテキストです。このアプローチは、可視インターフェイスを煩雑にせずにインタラクティブ要素に追加コンテキスト、説明、または説明的なラベルを提供するためによく適用されます。

開発者は、この効果を達成するために `.sr-only`、`.visually-hidden`、または `.element-invisible` などの一般的なCSSクラスをよく使用します。これらのクラスは通常、テキストをアクセシビリティツリーに残してスクリーンリーダーで読み取れるようにしながら、画面外の配置、クリッピング、またはゼロサイズのボックスを使用して視覚的にテキストを非表示にします。

図6.28. sr-onlyまたはvisually-hiddenクラスを持つデスクトップウェブサイト。

これらの一般的なスクリーンリーダー専用クラスの使用は2024年から2025年の間で基本的に変わらずです。一部のウェブサイトは、セマンティックHTMLだけでは明らかでない方法でスクリーンリーダーユーザーにコンテキストを提供するために隠しテキストを含んでいます。

動的にレンダリングされたコンテンツ

ARIAライブリージョンは、動的に変化するコンテンツをアクセシブルにするために重要です。フォームバリデーションメッセージ、ステータス更新、ライブフィードなどのページコンテンツの更新をスクリーンリーダーに通知します。これらの更新はページの完全リロードなしに発生するため、ユーザーが中断なしに重要な情報を受け取るために必要です。

ロール	デスクトップ	モバイル	暗黙の aria-live 値
status	15.18%	14.51%	polite
alert	7.12%	6.74%	assertive
timer	0.91%	0.84%	off
log	0.61%	0.55%	polite
marquee	0.09%	0.10%	off

図6.29. ライブリージョンARIAロールとその暗黙の `aria-live` 値を持つページ。

2025年において、サイトの約33%が `aria-live` 属性を使用しており、2024年から4%増加しています。本質的に `aria-live` 値が `polite` である `status` の `role` 値の使用が約5%増加しました。これは、緊急でない更新をユーザーに通知するていねいな通知の広範な採用を示しています。

`alert`、`timer`、`log`、`marquee` などの追加のARIAロールも、事前定義された動作で暗黙の `aria-live` 属性を持ち、幅広いライブリージョンのユースケースを可能にします。

2025年のARIAライブリージョンの使用増加は、動的なコンテンツ更新を効果的に伝達する進歩を反映しており、最新のレスポンシブなウェブ体験とインタラクトするために支援技術に依存するユーザーをサポートしています。

ユーザーパーソナライゼーションウィジェットとオーバーレイ修復

アクセシビリティウィジェットとオーバーレイ修復ツールは、ウェブサイトのアクセシビリティを向上させるためのサードパーティスクリプトです。フォントサイズやコントラスト調整などのユーザーパーソナライゼーションオプションと、一般的なアクセシビリティ問題の自動修正を提供します。

これらのオーバーレイは多くの場合、迅速なコンプライアンス修正を約束しますが、手動のコードとデザイン変更を必要とする複雑なアクセシビリティの課題に対処するには不十分で

す。欧州障害フォーラム[196]は、このようなツールがユーザー自身の支援技術に干渉し、アクセシビリティを低下させてユーザーをイライラさせる競合を引き起こす可能性があると警告しています。

オーバーレイは一部の表面的な障壁を取り除き、追加のパーソナライゼーション機能を提供するのに役立ちますが、それに依存することで組織が適切なアクセシビリティへの投資をやめてしまうことが多くなります。オーバーレイは一般的に、根本的なコードの問題を修正することと比較して、使いやすさ、セキュリティ、パフォーマンスの欠点がより多いです。

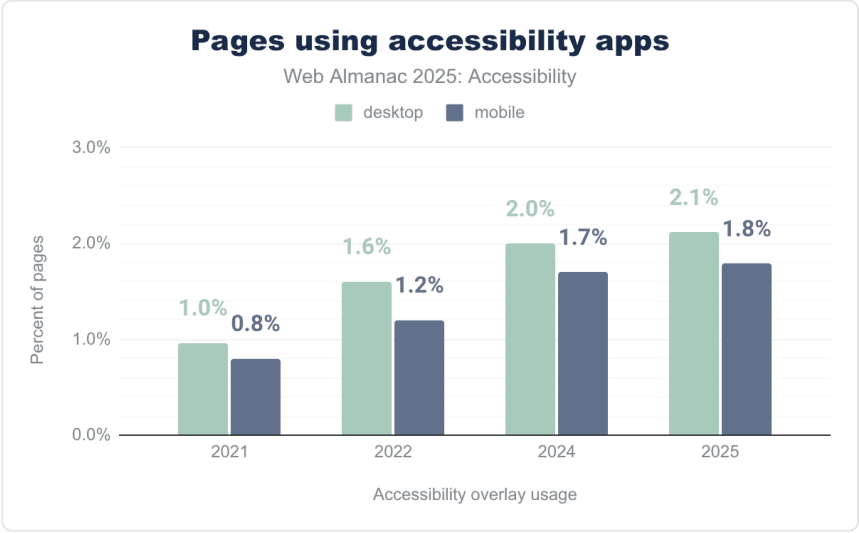


図6.30. アクセシビリティアプリ（オーバーレイ）を使用しているページ。

データによると、デスクトップサイトの約2%がそのようなアクセシビリティアプリを使用しています。最も高いトラフィックのサイトではさらに低く、トップ1,000では0.2%です。このパターンは、オーバーレイは主にトラフィックの少ないサイトに採用されており、議論の多い不完全なソリューションであることを示しています。

2025年の使用率の小幅な増加にもかかわらず、これらのアクセシビリティアプリの分布は2024年と一致しており、UserWay[197]、AccessiBe[198]、AudioEye[199]、EqualWeb[200] などのプロバイダーが主流です。

196. https://www.edf-feph.org/accessibility-overlays-dont-guarantee-compliance-with-european-legislation/
197. https://userway.org/
198. https://accessibe.com/
199. https://www.audioeye.com/
200. https://www.equalweb.com/

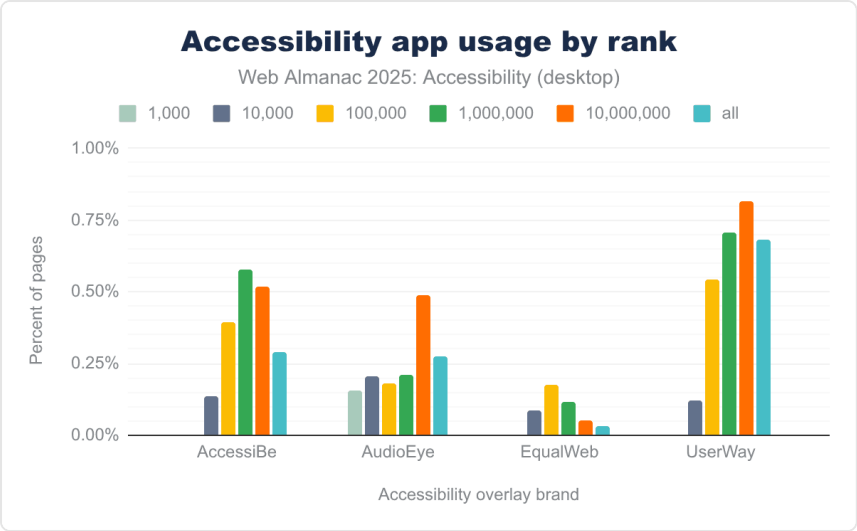


図6.31. ランク別アクセシビリティアプリの使用状況。

オーバーレイに関する混乱

アクセシビリティオーバーレイとパーソナライゼーションウィジェットは、2025年に約2%の小幅な使用増加を見せているものの、依然として大きな論争と混乱の源となっています。

国際アクセシビリティ専門家協会（IAAP）や欧州障害フォーラム（EDF）などの主要組織は、オーバーレイが万能薬ではないと明示的に警告しています。ユーザーのアクセスを妨げたり、支援技術に干渉したりしてはならず、サイトを完全に準拠させるものとしてマーケティングされるべきではありません。

マーケティングの主張は組織に非現実的な期待を生むことが多く、法的および実際的なリスクにつながります。これらのツールは、インクルーシブデザイン、手動アクセシビリティテスト、継続的な修復に代わることはできません。これらはすべてアクセシビリティ基準を満たすために不可欠です。

欧州アクセシビリティ法やその他の規制は、アクセシビリティはウェブサイトのソースに組み込まれなければならないことを強調しています。アクセシビリティオーバーレイが存在していても、基礎となるウェブサイト自体がアクセシブルでなければ、アクセシビリティ基準の詳細な基準を満たしません。したがって、それ自体では適切なソリューションではありません。

Bitvtest.deは最近、オーバーレイツールを使用するウェブサイトのテストを中止し、認証マ

ークの付与を禁止することを決定[201]しました。Bitvtestは、ウェブサイトやアプリのアクセシビリティをチェックするための構造化されたアプローチであるドイツのBITV-Testを基盤とした専門的なアクセシビリティテストサービスです。

人工知能（AI）の影響

人工知能（AI）は、通常人間の知能を必要とするタスクを実行できるシステムを構築することに焦点を当てたコンピュータサイエンスの広大な分野です。この分野の中で、大規模言語モデル（LLM）は統計的パターンを学習するために非常に大きなデータセットでトレーニングされた特定の種類のAIシステムです。

LLMはこれらのパターンを使用してテキストを生成・変換し、質問に答え、指示に従いますが、人間的な意味での理解や意図はありません。トレーニングデータに基づいて可能性の高い単語や文章を予測します。

ウェブアクセシビリティツールはLLMにますます依存しています。多くのプラットフォームが不足している画像説明に対処する方法としてAI生成の `alt` テキストを提供するようになりましたが、これらの自動化ソリューションの品質と精度は大きく異なります。

開発者はGitHubのアクセシビリティスキャナー[202]などのAIツールを日常業務に取り入れています。これらのツールはインスタントフィードバックとアクセシビリティの推奨事項を提供し、アクセシビリティの問題を修正しやすくします。

しかし、アクセシビリティにおけるAIには問題がないわけではありません。

現時点では、AIがウェブサイトやそのコンテンツを作成または改善したかどうかを確認する標準的な方法がありません。これにより、サイトの評価が難しくなり、ユーザーが何を見ているかわからないままにしています。

アクセシビリティアドボケートのLéonie Watsonのような専門家は、AIがオンラインでコンテンツとのインタラクション方法を変える「エージェンティックウェブ[203]」について話しています。これはアクセシビリティ基準をどのように適応させる必要があるかという疑問を提起します。

AI搭載のブラウザと拡張機能は急速に主流になりつつあります。ブラウザに組み込まれた音声アシスタントとAIエージェントは、基本的な情報検索のほとんどを間もなく処理するかもしれません。人々はすでに従来の検索結果の代わりにAI生成の回答を選択するオプションがあります。このシフトはアクセシブルなデザインを助けたり傷つけたりする可能性があります。それはすべてこれらのAIツール自体が完全にアクセシブルかどうかにかかっています。

201. *https://bitvtest.de/test-methodik/web/beschreibung-des-pruefverfahrens#c761*
202. *https://github.com/github/accessibility-scanner*
203. *https://tetralogical.com/blog/2025/08/08/accessibility-and-the-agentic-web/*

Joe Dolsonのような専門家はAIが独自に完全にアクセシブルなウェブサイトを構築できるかどうか[204]を探求し、技術の潜在的可能性と現在の限界の両方を強調しています。Scott VinkleのShopifyでの経験[205]は、AIが実際の状況でアクセシビリティを向上させる方法を示しています。

A11Y CollectiveブログのAIとアクセシビリティについて[206]は、`alt`テキストやリアルタイムキャプション、音声アシスタントを自動化するAIツールは規模でアクセシビリティを助けることができますが、精度、コンテキスト、プライバシー、バイアスにまだ苦労していると指摘しています。

Dries Buytaertの研究[207]は、AIが未ラベルの画像の膨大なバックログに対処できることを示していますが、品質のための人間によるレビューはまだ不可欠です。彼はAI搭載のalt textを検討している組織のための品質、プライバシー、コスト、複雑さのバランスを探ります。

デジタルアクセシビリティトレーニング[208]は、コンテンツクリエイター向けのAIの機会と課題を概説しています。AIツールは規模でアクセシビリティ機能を実現しますが、コンテンツの妥当性、バイアス、倫理的使用についての懸念を引き起こします。

Hidde de Vriesは人間と言語モデルがアクセシブルなコンポーネントコードにアプローチする方法を対比[209]しています。人間はインターフェイスの意図に導かれ、仕様、ユーザーニーズ、支援技術の動作、プラットフォームの特性に基づいてHTML、CSS、ARIAの決定を行います。LLMの代わりにトレーニングデータから可能性の高いコードを予測しますが、これは既存のコードのほとんどにアクセシビリティの問題があり、モデルには特定のユーザーへの意図や理解がないため問題です。

Adrian Roselliは、コンピュータビジョンとLLMの最近の進歩が、複雑な記事を理解しやすい要約に蒸留するのを読者が助ける可能性があることを認めています。しかし、彼はこれらのツールにはまだコンテキストと著者性が欠けている[210]と主張しています。コンテンツがなぜ作成されたか、ジョークやミームが何に依存しているか、またはインターフェイスがどのように機能することを意図されているかを知ることができません。彼らの説明とコードの提案は容易に要点を外れたり、ユーザーを誤解させたりする可能性があります。

AIはアクセシビリティを超えた重要な倫理的懸念を引き起こします。

204. *https://wpbuilds.com/accessibility/5/*
205. *https://scottvinkle.com/blogs/work/4-ways-i-use-ai-as-an-accessibility-specialist*
206. *https://www.a11y-collective.com/blog/artificial-intelligence-accessibility/*
207. *https://dri.es/comparing-local-llms-for-alt-text-generation*
208. *https://www.digitalaccesstraining.com/pages/articles?p=ai-and-accessibility-opportunities-and-challenges-for-content-creators*
209. *https://hidde.blog/ai-for-accessible-components/*
210. *https://adrianroselli.com/2023/06/no-ai-will-not-fix-accessibility.html*

図6.32. LLMのトレーニングに必要な炭素排出量に相当するガソリン車の台数。

主要な問題の1つは環境への影響です。炭素排出量と大規模ニューラルネットワークのトレーニングに関する研究[211]は、GPT-3のような単一の大規模言語モデルのトレーニングが1,200メガワット時以上の電力を消費し、約500〜550メートルトンのCO_2相当を生産したと推定しています。これは100台以上のガソリン車の生涯排出量に相当します。詳しい情報についてはサステナビリティチャプターを参照してください。

もう1つの核心的な懸念は、トレーニングデータがどのように収集・使用されるかです。高プロファイルな訴訟[212]のいくつかは、AIベンダーが許可や補償なしに著作権のある素材をスクレイピングして使用したと主張しています。これには何百万ものストック写真や本が含まれていました。

大規模モデルの研究[213]は、それらがトレーニングデータに存在する障害差別的、人種差別的、その他の有害なバイアスを増幅できることを示しています。これは最終的に、障害、人種、性別の説明方法を形成し、規模でスティグマを強化します。

先を見据えると、AIはウェブ開発者とコンテンツクリエイターが依存するツールとしてますます重要になることは明らかです。しかし、AIは人間の専門知識を支援すべきであり、置き換えるものではありません。

これらのツールが改善されるにつれて、アクセシビリティコミュニティは2026年以降のいくつかの重要な問題に答える必要があります：

- AI生成コンテンツが正確でアクセシブルかどうかを確認するにはどうすればよいか？
- ウェブコンテンツにおけるAIの使用を管理すべき基準は何か？
- 支援技術はAI駆動インターフェイスとどのように機能するか？
- AIは個人のニーズを尊重しながら平等なアクセシビリティを提供できるか？
- AIがバイアスを強化したり人々を排除するのを防ぐためにはどのような倫理ガイドラインが必要か？

211. *https://arxiv.org/ftp/arxiv/papers/2104/2104.10350.pdf*
212. *https://natlawreview.com/article/two-major-lawsuits-aim-answer-multi-billion-dollar-question-can-ai-train-your*
213. *https://www.boia.org/blog/ethics-of-ai-in-accessibility-avoiding-bias-when-using-generative-ai-text*

セクターとアクセシビリティ

このセクションでは、パターンとリーダーを特定するために、さまざまな業界およびコミュニティセクター間でアクセシビリティスコアを比較します。

国

ウェブサイトの原産国は、サーバーの地理的位置（GeoID）またはトップレベルドメイン（TLD）によって特定できます。どちらの方法にも限界があります。ホスティングコスト、サーバーロケーション戦略、ドメイン所有慣行により、ウェブサイトのサーバーが対象ユーザーを反映していない場合があります。`.ai` や `.io` などのグローバルに使用されているTLDは、必ずしも原産国に結びついているわけではありません。

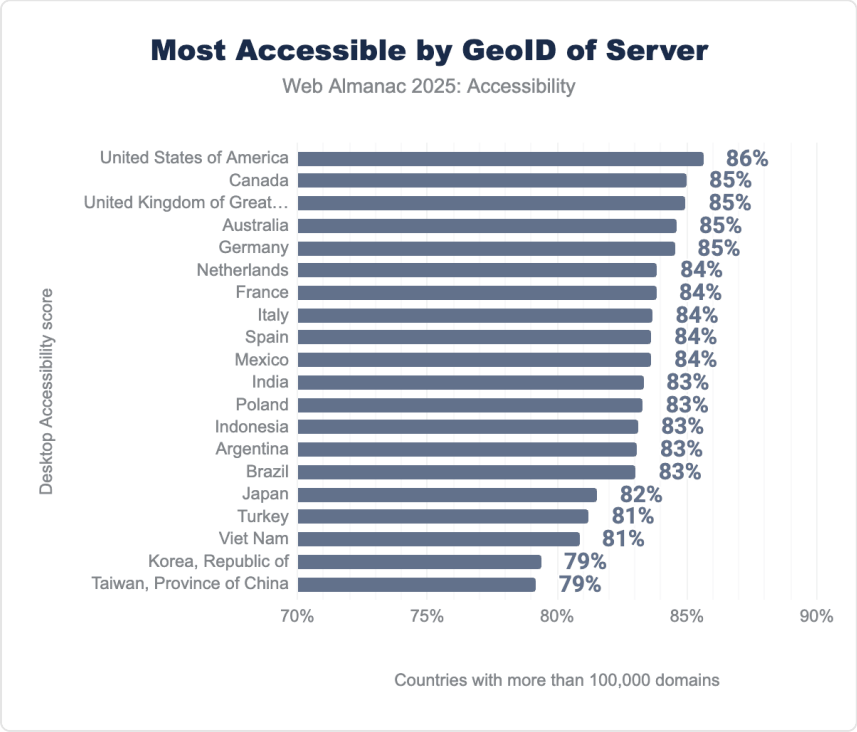


図6.33. サーバーのGeoIDによる最もアクセシブルな国

2025年、米国はGeoIDによる最もアクセシブルな国として引き続きトップの位置を維持しており、これは連邦機関に対する数十年にわたるSection 508コンプライアンス要件とADA第III編訴訟の継続によって推進されています。`.edu` と `.gov` TLDもアクセシビリティ指標をリ

ードしており、米国政府および教育機関の必須コンプライアンスを反映しています。

2025年におけるEUアクセシビリティ法の限定的な影響も注目されます。この法律は2025年6月28日に完全施行され、欧州連合全域で民間・公共セクターのデジタルサービスをアクセシブルにすることを義務付けましたが、予備データはヨーロッパベースのサイトのウェブアクセシビリティに劇的な上昇がないことを示しています。

このラグは、実装上の課題、既存サービスの移行期間、および組織がデジタルオファリングを再設計・監査するために必要な時間を反映していると考えられます。

法的執行と訴訟の脅威は、米国のリーディングポジションが示すように、アクセシビリティコンプライアンスの最も強力な推進力であり続けています。新しいヨーロッパおよびグローバルな法律の完全な影響が、組織が実装スケジュールと移行期間を調整する中で、ウェブアクセシビリティ統計に現れるには数年かかる可能性があります。

2025年に注目すべきトレンドが現れました。`.ai` ドメインがアクセシビリティランキングに初めて登場し、現在は `.edu` と `.gov` を除くすべてのTLDを上回っています。これは、アクセシビリティを含む最新の開発慣行を優先するAI関連ビジネスの採用拡大を反映していると考えられます。

1995年に小さなカリブ海の島であるアンギラ[214]に最初に割り当てられた `.ai` TLD拡張子は、2009年にアンギラが全世界への直接登録を開放するまで、15年近くほとんど無名のままでした。このドメインは、2022年以降の人工知能ブームがそれをグローバルで最も急成長するTLDの1つに変えるまで、その歴史のほとんどを休眠状態で過ごしました。

転機となったのは、2022年後半のChatGPTの登場で、AIへの前例のない関心を呼び起こしました。2022年7月から2023年7月の間に、登録済みの `.ai` ドメインは急増し、75,314件から196,292件へと、わずか12ヶ月で161%増加しました。

地理的には、北米が `.ai` 登録[215]の62.5%を占め、米国だけで全登録済み `.ai` ドメインの62.5%を占めています。アジアが18.8%、ヨーロッパが17.2%で続いています。

多くの `.ai` ドメイン保有者はベンチャー支援を受けた、資金力のあるテクノロジースタートアップであり、AIを使用する可能性が高いです。AIが人間チームよりもアクセシブルなコードを生成できるかどうかはまだ議論中ですが、これは1つの指標です。

`.com` や `.org` を使用する古い伝統的な企業とは異なり、新しいAI企業は最初からアクセシビリティを考慮した現代のウェブ標準とツールで構築することが多いです。`.ai` ドメインを使用しAI製品を販売するこれらの企業は、少なくともコードやコンテンツのために相談するという形で、より自動化されたエージェント的なセットアップで使用していない場合でも、ウェブサイト構築にAIを関与させている可能性が非常に高いです。

214. *https://en.wikipedia.org/wiki/Anguilla*
215. *https://techjury.net/industry-analysis/the-rise-of-ai-domains/*

`.com`、`.org`、`.net` などの伝統的なTLDはアクセシビリティのリーダーとしてランク付けされていません。これは、ドメインタイプだけではアクセシビリティコンプライアンスの強力な予測因子ではないことを示唆しています。

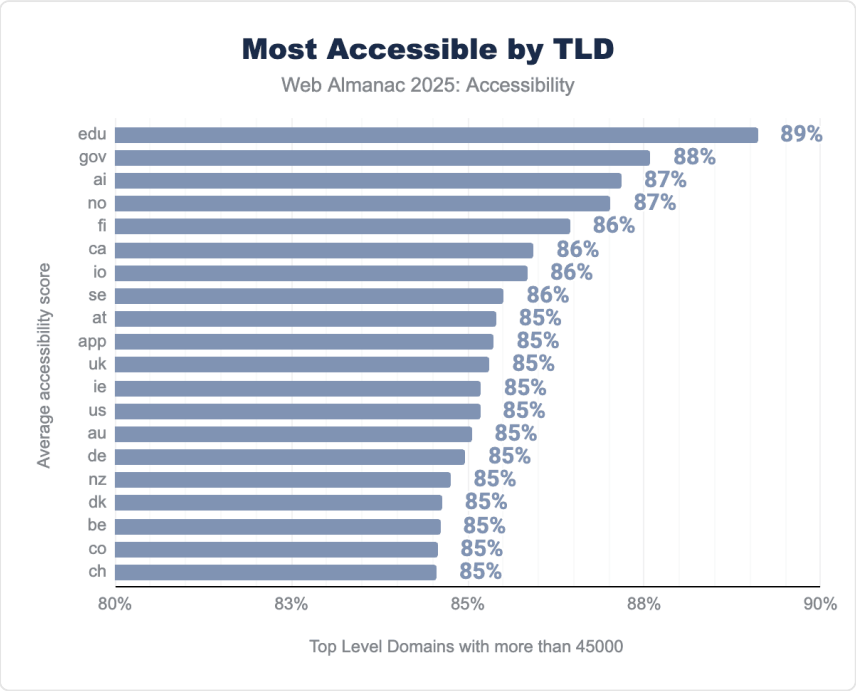


図6.34. トップレベルドメイン（TLD）によるアクセシブルな国。

TLDランキングの地図は2024年と非常に似ていますが、現在利用可能な非国特定TLDの増加数は当然ながら含まれていません。

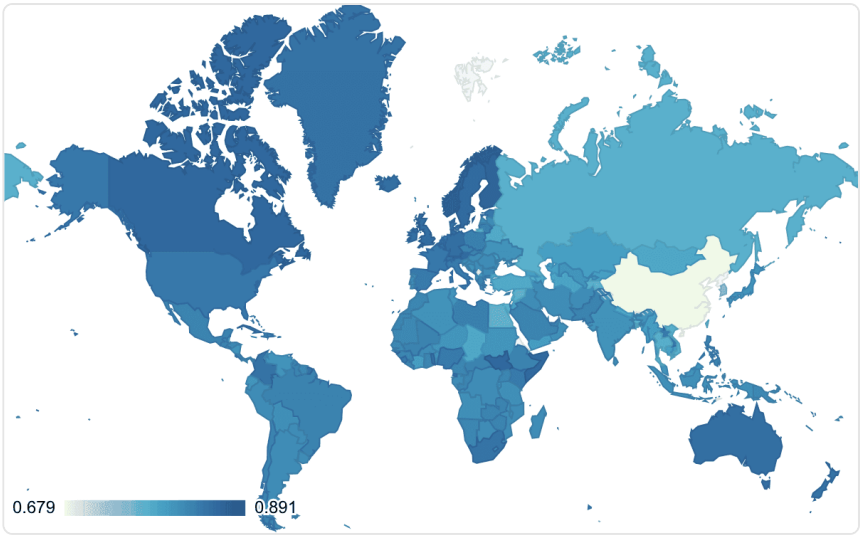


図6.35. トップレベルドメイン（TLD）によるアクセシブルな国の地図。

政府

政府ウェブサイトは、アクセシビリティへの公的コミットメントを示す重要な場であり続けています。しかし、実装は各管轄区域によって大きく異なります。2025年のデータは、最近の法律、データ収集の方法論的変化、執行メカニズムの影響を受けたグローバルな政府ウェブサイトアクセシビリティの重要なトレンドを明らかにしています。

今年は、2024年のスキャンに含まれなかったより広範なドメインを評価するために、異なる方法論を使用しました。

2024年には、オランダ政府から79ドメインのみをサンプリングしました。2025年には957ドメインと10倍以上の数を照会しました。同様に、ルクセンブルクとフィンランドについては約2倍の数のドメインをスキャンしました。精度の向上により、より包括的なデータセットが得られますが、前年比較も複雑になります。

英国では、アクセシビリティが94%に改善し、2024年から2%上昇しました。これは、すべての公共デジタルサービスにおけるアクセシビリティを優先する英国政府のデジタルサービス基準[216]などの標準化されたデザインシステムの恩恵を反映しています。オランダ、ルクセンブルク、フィンランドは引き続きリードしており、オランダは過去の年に98%を達成しています。これらの国は、一貫したガバナンスフレームワークとデザインシステムの優先化によってこの地位を維持しています。

216. *https://www.gov.uk/service-manual/service-standard*

スコットランド（gov.scot）とウェールズ（gov.wales）も含めるよう努力しました。2025年には19,568ドメインから平均を算出しましたが、2024年は16,594ドメインのみでした。

モニタリングはアクセシビリティを優先するための重要な要素であり、アクセシビリティを含む多くのウェブサイト品質指標の進捗状況を強調するフランス政府のダッシュボード[217]のような取り組みを称賛します。この背後にあるコードの多くはオープンソースです。オランダ政府も約9,000のウェブサイトとアプリケーションを追跡するダッシュボード[218]を持っています。執筆時点で、追跡されているすべてのウェブプロパティの60%が法的要件に準拠しています。

AccessibleEUが実施するアクセシビリティモニタリングレポート[219]は重要ですが、より抽象的です。EUのウェブアクセシビリティ指令[220]は、加盟国に3年ごとにアクセシビリティコンプライアンスを監視・報告することを要求しています。これらのレポート[221]は公開されています。

欧州連合の進化する規制環境は、2025年の政府ウェブサイトアクセシビリティに大きな影響を与えています。

EUウェブアクセシビリティ指令は、公共セクターの組織に特定の技術的基準を満たすことを要求しています。2025年6月28日に完全施行されたより広範な欧州アクセシビリティ法（EAA）は、eコマース、旅行、銀行などの主要セクターの民間セクター組織に要件を拡大しています。

この規制の勢いにもかかわらず、2025年のアクセシビリティデータは欧州政府ウェブサイトコンプライアンスに劇的な急増を示しておらず、実装がまだ進行中であり、完全な影響は2026年まで見えないかもしれないことを示唆しています。

EUの加盟国は、ユーザーがバリアを報告するためのアクセシビリティステートメントを公開し、フィードバックメカニズムを提供することが求められています。アクセシビリティステートメント[222]はEUのウェブアクセシビリティ指令の重要な部分ですが、現時点ではサイトスキャンに含める良い方法がありません。Funka Foundationは、コンプライアンスのためのこのタイプのテストの限界[223]について思い起こさせてくれています。

217. *https://observatoire.numerique.gouv.fr/observatoire*
218. *https://dashboard.digitoegankelijk.nl/*
219. *https://accessible-eu-centre.ec.europa.eu/accessibility-monitoring_en*
220. *https://en.wikipedia.org/wiki/Web_Accessibility_Directive*
221. *https://digital-strategy.ec.europa.eu/en/news/web-accessibility-directive-monitoring-reports-2022-2024*
222. *https://cerovac.com/a11y/2025/07/we-need-to-talk-about-your-accessibility-statement/*
223. *https://www.linkedin.com/pulse/yes-nordics-score-well-lets-think-one-step-further-funkafoundation-yczzf/*

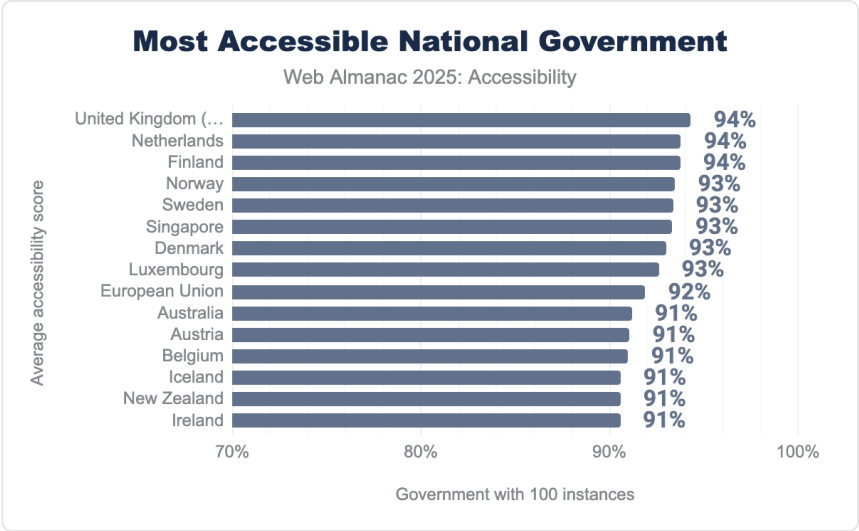


図6.36. 最もアクセシブルな国家政府。

2025年の地図は2024年とほとんど区別がつきません。

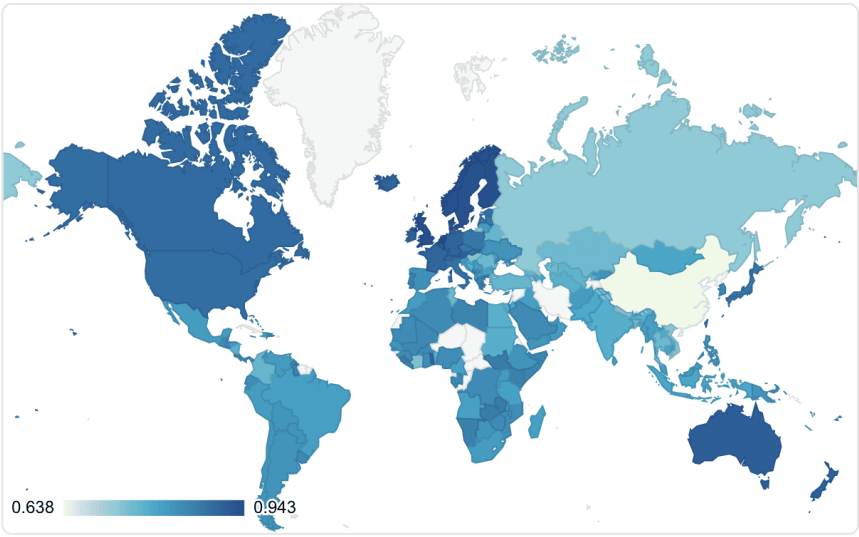


図6.37. グローバルな政府ウェブサイトのアクセシビリティの地図。

米国では、新しい連邦義務にもかかわらず、州政府のコンプライアンスは一貫性がないままです。2024年6月に公表された障害を持つアメリカ人法（ADA）第II編に関する司法省

（DOJ）の最終規則は、すべての州および地方政府機関がWCAG 2.1 AAコンプライアンスを特定の期限までに達成することを要求しています：人口50,000人以上を対象とする機関は2026年4月26日まで、小規模機関は2027年4月26日まで。

コロラドやバーモントなどの州は、集中型ガバナンス構造を確立することで優れた成果を上げています。コロラド州の全州インターネットポータル局[224]（SIPA）は、集中管理が複数の機関にわたるアクセシビリティを改善する方法を示しています。

ネバダ、カンザス、カリフォルニア、ニューヨークは両年のサンプルで好成績でした。しかし、平均値は州政府が2024年の司法省ウェブおよびモバイルアプリアクセス強化のための最終規則[225]の新要件の達成に有意な進歩を遂げたことを示していません。全国州最高情報責任者協会（NASCIO）全国会議での州技術リーダーたちは、アクセシビリティを最優先事項として再確認しました[226]。

7月に35周年を迎えた[227]障害を持つアメリカ人法（ADA）は、グローバルに前例を設けた先駆的な取り組みでした。

オープンソースツールOobee[228]を通じて示されたシンガポールのアクセシビリティ改善への最近のコミットメント[229]は、新興のグローバルモメンタムを示しています。Oobeeは組織が数百ページをスキャンし、統合アクセシビリティレポートを生成できるようにし、デジタル公共財[230]としての位置づけをしています。

224. *https://sipa.colorado.gov/*
225. *https://www.justice.gov/archives/opa/pr/justice-department-publish-final-rule-strengthen-web-and-mobile-app-access-people*
226. *https://statescoop.com/accessibility-nascio-state-priority-2025/*
227. *https://www.access-board.gov/news/2025/07/21/celebrating-35-years-of-americans-with-disabilities-act/*
228. *https://github.com/GovTechSG/oobee*
229. *https://archive.opengovasia.com/2025/03/11/digital-accessibility-singapores-commitment-to-inclusivity/?c=ca*
230. *https://www.digitalpublicgoods.net/r/oobee*

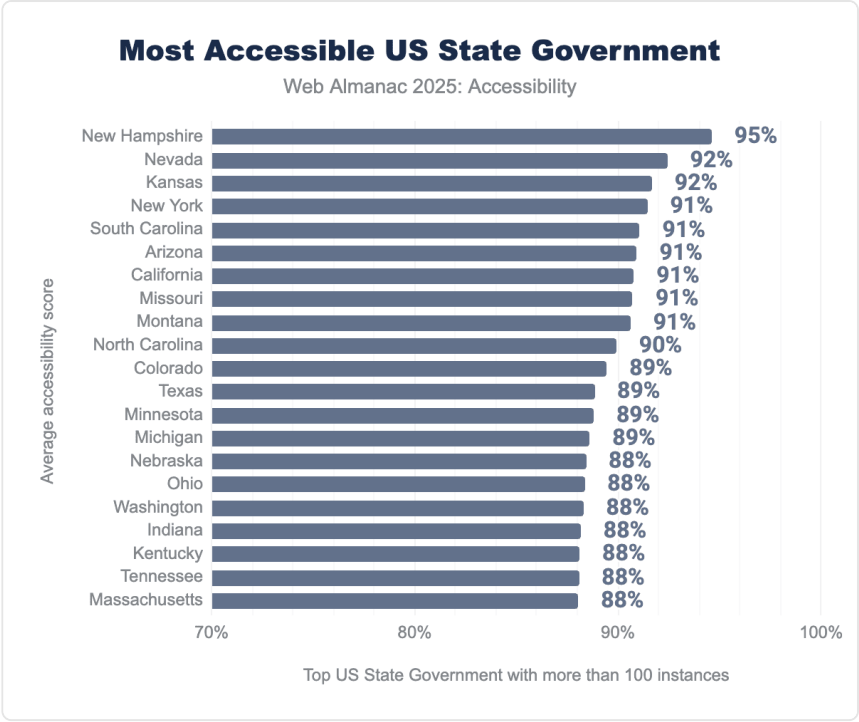


図6.38. 最もアクセシブルな米国州政府。

コンテンツ管理システム（CMS）

ウェブサイトのコンテンツ管理システム（CMS）の選択は、アクセシビリティの結果に大きな影響を与えます。Wappalyzer[231]のデータを使用して、2025年のWeb Almanacは従来のCMS、プラットフォーム、専門的なウェブサイトビルダー全体でアクセシビリティスコアを比較しました。これにより、一貫したパターンと注目すべき外れ値の両方が明らかになりました。

231. *https://almanac.httparchive.org/en/2024/methodology#wappalyzer*

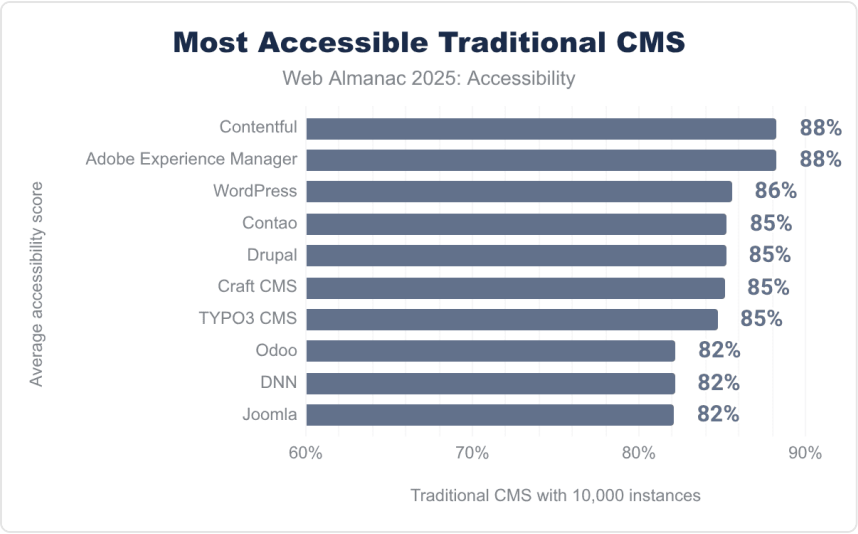


図6.39. 最もアクセシブルな従来のCMS。

従来のCMSの中で、Sitecore[232]は2025年に85%のアクセシビリティを維持しましたが、インスタンス数は10,000を下回りました。Adobe Experience Manager（AEM）[233]とContentful[234]は引き続きリードしており、これらのエンタープライズソリューションを採用する大企業がアクセシビリティの問題に対処するためのリソースをより多く持っているためと考えられます。WordPress[235]は2024年から有意な改善を示しませんでしたが、市場支配力とユーザーベースの高まるアクセシビリティ意識を反映して、3位に上昇しました。

注目すべきことに、上位5つの従来のCMSは一貫したエラーパターンを共有しています。カラーコントラスト、リンク名、見出し順序が最も一般的な問題として支配的です。CMSはリンクの命名や色の選択を指示できないため、これらのエラーは主にプラットフォームの制限ではなくコンテンツの選択を反映しています。ただし、AEMだけが上位5つのエラーに `label-content-name-mismatch` を含んでいます。WordPressは `meta-viewport` エラーを持つ点でユニークです。

上位10のCMSの中で、DNN[236]だけが上位3つのエラーに `image-alt` を持っています。ほとんどの従来のCMSでは、`image-alt` と `target-size` はGoogle Lighthouseエラーの4位または5位に一貫して位置しています。

232. https://www.sitecore.com/
233. https://business.adobe.com/products/experience-manager/sites/aem-sites.html
234. https://www.contentful.com/
235. https://wordpress.com/
236. https://www.dnnsoftware.com/

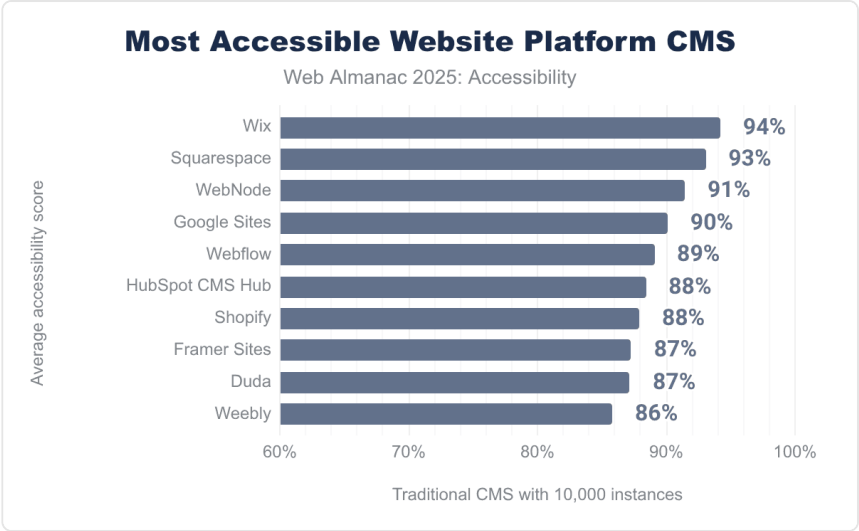


図6.40. 最もアクセシブルなウェブサイトプラットフォームCMS。

プラットフォームCMS	最も多い	2番目	3番目	4番目	5番目
Wix[237]	heading-order	link-name	color-contrast	button-name	target-size
Squarespace[238]	color-contrast	heading-order	link-name	label-content-name-mismatch	frame-title
Webnode[239]	heading-order	link-name	frame-title	color-contrast	image-redundant-alt
Google Sites[240]	image-alt	link-name	aria-allowed-attr	heading-order	color-contrast
Webflow[241]	link-name	color-contrast	heading-order	html-has-lang	target-size

図6.41. 人気のCMSプラットフォームの上位アクセシビリティ監査問題。

Wix、Squarespace、Google Sitesなどのウェブサイトプラットフォームは、アクセシビリティィにおいて従来のCMSを大幅に上回っています。この優れたパフォーマンスは、そのアプローチに起因していると考えられます。これらのプラットフォームは、テンプレート化された

237. https://www.wix.com/
238. https://www.squarespace.com/
239. https://www.webnode.com/
240. https://workspace.google.com/products/sites/
241. https://webflow.com/

デザインと組み込みのアクセシビリティデフォルトを通じてユーザーの選択を制限し、アクセシビリティの悪い決定の機会を減らすことがよくあります。

データは、コンテンツクリエイターが一部の決定に最終的な責任を持つ場合でも、CMSの選択がアクセシビリティの結果に有意な影響を与えることを証明しています。より厳格なデザイン制約と組み込みのアクセシビリティデフォルトを持つプラットフォームはより優れたパフォーマンスを示し、最大の柔軟性を提供するプラットフォームはアクセシビリティの決定をユーザーに委ねます。

JavaScriptフロントエンドフレームワーク

JavaScriptフレームワークの選択も、ウェブサイトのアクセシビリティの結果に大きな影響を与えます。State of JSレポート[242]の分類を使用して、さまざまなUIフレームワークとメタフレームワークがアクセシビリティパフォーマンスとどのように相関するかを調べました。これにより、パターン、変化、そして新興の懸念が明らかになりました。

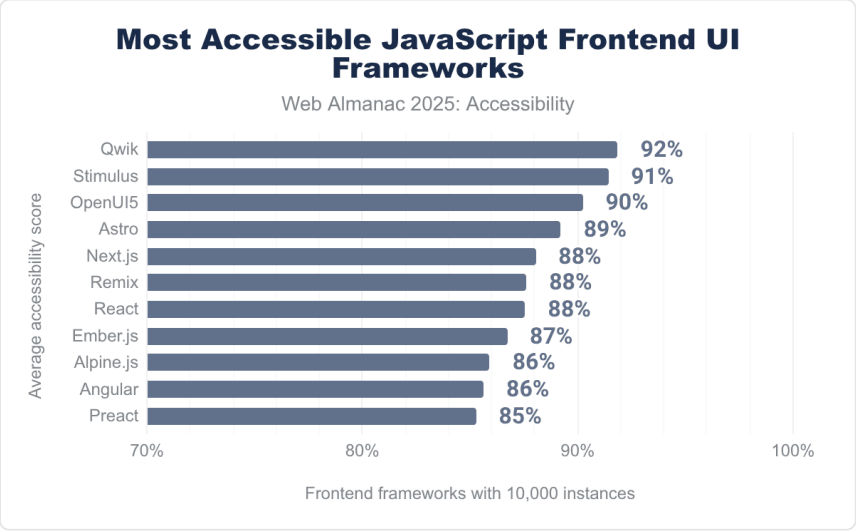


図6.42. 最もアクセシブルなJavaScriptフロントエンドUIフレームワーク。

2025年、OpenUI5[243]がアクセシビリティランキングで上昇した一方、2024年にリードしていたフレームワーク（Stimulus[244]、Remix[245]、Qwik[246]）は順位が変動しました。RemixはUIとメタフレームワークの両方のカテゴリに登場しますが、2025年にはランキングが低下し、

242. *https://stateofjs.com/en-US*
243. *https://openui5.org/*
244. *https://stimulus.hotwired.dev/*
245. *https://remix.run/*
246. *https://qwik.dev/*

他のフレームワークが前進できるようになりました。この不安定さは、サンプルサイズの変化や競合フレームワークの実際の改善を反映している可能性があります。Remixのすべての機能がReact Router[247]の次のバージョンに統合され、Remixがリアクトフレームワークとして冗長になったという事実も、ランキングに影響を与える可能性があります。

歴史的に、Stimulus、Remix、Qwikは、プログレッシブエンハンスメントとセマンティックHTMLを優先しているため、React、Svelte[248]、Ember.js[249]などのメインストリームオプションを数パーセントポイント上回っていました。

メタフレームワークの中では、Remix、RedwoodJS[250]、Astro[251]が2024年にリードしており、Remixの低下によりGatsby[252]が2025年に3位に上昇しました。サーバーファーストのメタフレームワーク（SvelteKit、Astro、Remix、Qwik、Fresh[253]、Analog[254]）の台頭は、クライアントサイドJavaScriptの複雑さを軽減することで、より優れたパフォーマンスとアクセシビリティの実践に向けた広範な業界のシフトを反映しています。

ライブラリの選択もアクセシビリティに影響します。

Reactは最大の柔軟性とカスタマイズを提供しますが、開発者がアクセシビリティを意図的に実装することを要求します。その広範なエコシステムにはReact Aria[255]やReach UI[256]などのアクセシビリティ重視のライブラリが含まれていますが、アクセシビリティはデフォルトで強制されません。

Angular[257]は、強力な組み込みアクセシビリティ機能、セマンティックHTMLを促進する構造化された規則、ARIA属性サポート、およびキーボードナビゲーションとスクリーンリーダーサポートを標準搭載したMaterial Design[258]コンポーネントを提供します。Angularの意見が強い構造は、開発者をより標準化されたアクセシブルな実践に向けて導く傾向があります。

Vue.js[259]は、Reactの柔軟性とAngularの構造のバランスを目指しています。Vueのプログレッシブデザイン、明確なテンプレート構文、コンポーネントアーキテクチャはアクセシビリティをサポートしていますが、vue-a11y[260]などのサードパーティプラグインへの依存度が高くなっています。

GitHubがグローバルアクセシビリティ啓発デー（GAAD）の誓約[261]を取り、オープンソースのアクセシビリティを大規模に改善することも注目されます。このコミットメントは重要なギャップに対処しています：企業の90%がオープンソースを使用し、コードベースの97%が

247. https://reactrouter.com/
248. https://svelte.dev/
249. http://Ember.js
250. https://rwsdk.com/
251. https://astro.build/
252. https://www.gatsbyjs.com/
253. https://fresh.deno.dev/
254. https://analogjs.org/
255. https://react-spectrum.adobe.com/react-aria/index.html
256. https://reach.tech/
257. https://angular.dev/
258. https://material.angular.io/
259. http://Vue.js
260. https://github.com/vue-a11y
261. https://github.blog/open-source/social-impact/our-pledge-to-help-improve-the-accessibility-of-open-source-software-at-scale/

オープンソースコンポーネントを含み、商業ツール内のコードの推定70〜90%がオープンソースに由来しています。

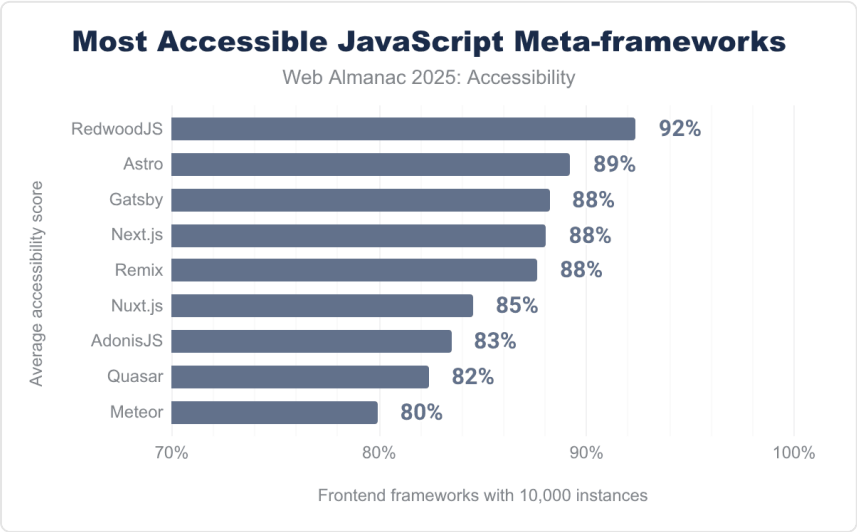


図6.43. 最もアクセシブルなJavaScriptメタフレームワーク。

RedwoodJSが最もアクセシブルであり、AstroとGatsbyが続いています。

結論

2025年のWeb Almanacアクセシビリティ分析は、重大な課題と機会の喪失によって抑制された、段階的な進歩の1年を明らかにしています。

自動テストは、規模を問わずアクセシビリティを評価するために不可欠なままです。しかし、データは測定だけでは意味のある改善を保証しない[262]ことを示しています。ウェブコミュニティは、障害を持つ何百万ものユーザーを引き続き排除しているシステム上の問題に対処するために、基本的なコンプライアンス指標を超えなければなりません。

2025年の最も顕著な改善は、規制圧力と執行メカニズムが最も強いセクターと地域で生まれました。しかし、2025年6月28日の欧州アクセシビリティ法の期限後に劇的な改善がなかったことは示唆的です。完全な影響は2026年以降まで明らかにならないかもしれません。

`.ai` ドメインが最もアクセシブルなTLDの1つに急速に台頭したことは、重要なパターンを反映しています。新興のベンチャー支援テクノロジー企業は、最初からモダンなアクセシビ

262. *https://www.a11y-collective.com/blog/how-to-check-web-accessibility/*

リティの実践で構築する傾向があり、一方でレガシーウェブサイトはアクセシブルでないままであることが多いです。これは、開発初期に優先された場合、アクセシビリティが達成可能であることを証明しています。

特定の分野での改善にもかかわらず、2024年に特定されたコアのアクセシビリティバリアは2025年でもほとんど変わらず持続しています。

カラーコントラスト、リンクの命名、見出し階層、および画像の `alt` テキストは、ほぼすべてのプラットフォームとフレームワークで上位4つの問題として残っています。これらは、ツールの直接的な失敗ではなく、著者がそれらをどのように使用するかの問題です。W3Cのアクセシビリティの役割と責任マッピング（ARRM）[263]によると、著者はこれらすべての失敗において何らかの役割を担っています。

この現実は重要な洞察を明らかにしています。CMSプラットフォーム、JavaScriptフレームワーク、ウェブテクノロジーはアクセシビリティの基盤を提供できますが、コンテンツクリエイターにアクセシブルな選択を強制することはできません。オーサリングツールアクセシビリティガイドライン（ATAG）2.0[264]や新しいW3CのATAGコミュニティグループ[265]のようなアプローチが役立つかもしれません。

2025年の指標は、段階的な改善が期待された場所での停滞を示唆しており、測定しやすいものと修正しやすいものの間のギャップを浮き彫りにしています。

アクセシビリティオーバーレイの継続的な台頭（現在はサイトの2%）は懸念されます。組織が真のアクセシビリティよりもショートカットを選ぶことが多いように見えます。IAAPと欧州障害フォーラムは、オーバーレイがユーザーの支援技術を妨げる可能性があり、アクセシブルなデザインの代替として使用してはならないと明示的に警告しています。2025年のデータは、オーバーレイが低トラフィックサイトに集中したままであることを確認しており、これは高品質でリソースが豊富な組織が真の解決策に向けてそれらから離れつつあるサインです。

2025年のデータは、自動化が必要だが不十分であることを強調しています。Lighthouseおよび類似ツールは簡単に測定可能な違反を検出しますが、ウェブ上の画像の50%は空または不十分な `alt` テキストを持っています。見出し階層は監査できますが、意味的な意味のある性質は人間の判断を必要とします。カラーコントラストは確認できますが、視覚的なデザインの選択はアクセシビリティ要件に基づいた主観的な芸術的決断を伴います。

2025年の調査結果は、障害を持つ何百万もの人々にとって依然として大部分がアクセス不可能なウェブを明らかにしています。

特定の分野での段階的な改善は励みになりますが、カラーコントラスト、リンクの命名、見出し構造、画像の説明の持続的なギャップは、ウェブコミュニティがまだアクセシビリティ

263. https://www.w3.org/WAI/planning/arrm/tasks/
264. https://www.w3.org/WAI/standards-guidelines/atag/
265. https://www.w3.org/community/atag/

を真の優先事項にしていないことを示しています。

`.ai` ドメインの台頭、GitHubのオープンソースへの誓約、EUアクセシビリティ法やADA第II編の最終規則などの規制期限は、2026年にはより実質的な変化が見られるかもしれないという希望を与えています。それは、組織が測定から説明責任へ、口先だけのことからリソースへ、コンプライアンスから真のインクルージョンへと移行する場合に限ります。

ウェブはすべての人のために機能すべきです。その原則がデザイン、開発、デプロイメントの決定を導くまで、このレポートに記録されたアクセシビリティのギャップは持続し続けるでしょう。

著者

Bogdan Lazar

@bogdanlazar.com tricinel tricinel https://bogdanlazar.com

Bogdanは、進行中の作業を妨げることなくアクセシブルなウェブサイトを構築する支援をするアクセシビリティコンサルタントです。教育・ヘルスケア分野での10年以上の経験を持ち、インクルーシブデザインとウェブアクセシビリティの深い専門知識を有しています。2024年3月から「誰もが入れる場所を作る」という信念のもと、アクセシビリティに関する日刊ニュースレター[266]を執筆しています。

Mike Gifford

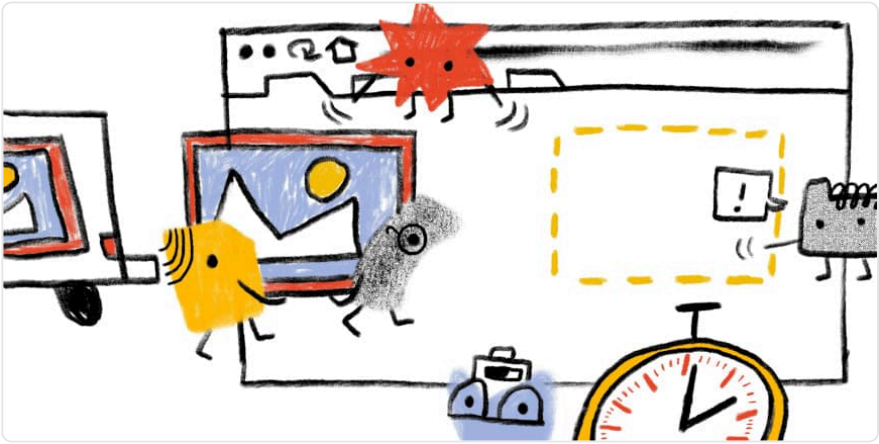
@https://mastodon.social/@mgifford @mgifford.bsky.social mgifford mgifford

https://accessibility.civicactions.com/

Mike GiffordはCivicActionsのオープンスタンダード＆プラクティスリードです。オープンガバメント、デジタルアクセシビリティ、サステナビリティに関するソートリーダーでもあります。Drupalコアアクセシビリティメンテナーおよびw3C招聘エキスパートを務めており、オーサリングツールアクセシビリティの専門家として認められ、W3Cのドラフト版ウェブサステナビリティガイドライン（WSG）1.0にも貢献しています。

266. *https://dailyaccessibility.com*

部 II 章 7
パフォーマンス



Himanshu Jariyal、 Prathamesh Rasam、 Humaira と Aaron T. Grogg によって書かれた。
Barry Pollard、 Stoyan Stefanov と Tanner Hodges によってレビュ。
Tanner Hodges による分析。
Barry Pollard 編集。
Sakae Kotaro によって翻訳された。

はじめに

ウェブパフォーマンスとは、ウェブページがどれだけ速くスムーズに読み込まれ、ユーザーの操作に反応するかを指します。パフォーマンスは、特にさまざまなデバイスやネットワーク環境でウェブが使用される中で、エンゲージメント、リテンション、全体的な信頼感を形成する上で重要な役割を果たします。速くレスポンシブに感じられるページは探索や継続的な利用を促す一方、遅かったり予測不可能な体験はフローを妨げ、信頼感を低下させます。したがって、パフォーマンスに影響する要因を理解することは、エンドユーザーにとって信頼できると感じるウェブ体験を構築するために不可欠です。

ウェブパフォーマンスの計測には、実際の環境でページがどのように読み込まれ、レンダリングされ、ユーザー入力に反応するかを表す広範な指標が含まれます。デバイス、ネットワーク、実行の制約により、ウェブが常に瞬時に感じられるとは限りません。そのため、パフォーマンスは速度だけでなく、処理が進んでいる間の体験の感触にも関係します。コンテンツが読み込まれる間に明確なフィードバックを提供し、レイアウトを視覚的に安定させるこ

とで、ユーザーはページの動作を理解し、ウェブサイトを操作する際にコントロールしている感覚を得られます。

これらの考慮事項が、Core Web Vitalsと呼ばれるユーザー中心のパフォーマンス指標[267]の開発と採用に影響を与えました。具体的には、読み込みパフォーマンス、応答性、視覚的安定性の主要な側面を捉えるLargest Contentful Paint（LCP）[268]、Interaction to Next Paint（INP）[269]、Cumulative Layout Shift（CLS）[270]です。過去1年間で、Core Web VitalsのうちLCPとINPの2つについて、報告のサポートがChrome以外のブラウザにも拡大[271]され、ブラウザエンジン間でユーザー体験をより一貫して計測できるようになりました。

これらの指標は、Time to First Byte（TTFB）[272]やFirst Contentful Paint（FCP）[273]などの従来の指標、およびページリソースの読み込み動作の計測によって補完されています。これらの広範なシグナルの集合は、パフォーマンスのボトルネックがどこで発生しやすいか、そしてそれが全体的なページ動作とどのように関係するかを説明するのに役立ちます。最新のウェブパフォーマンス指標のより包括的な概要はweb.dev[274]で確認できます。

このパフォーマンスチャプターでは、デバイスとネットワーク環境をまたいで大規模にこれらのシグナルを調査し、ウェブパフォーマンスの現状をデータに基づいた視点で提供します。実際のデータを分析することで、ウェブが改善されている箇所、課題が残る箇所、そしてより良いユーザー体験と関連するパターンを明らかにします。今年の分析には、Early Hints[275]やSpeculation Rules[276]などの新興パフォーマンス機能も含まれています。

データソースと方法論

この章は、HTTP Archive[277]とChrome UX Report（CrUX）[278]のデータをもとに、ラボベースの計測と実ユーザーのパフォーマンスデータを組み合わせています。HTTP ArchiveはWebPageTest経由でChromeベースのページ読み込みデータを収集し、制御された条件下でのページ動作についての詳細な洞察を提供します。一方CrUXは、Chromeユーザーから収集した実際のユーザー体験を反映しています。主な分析は2025年7月の計測に基づいており、ウェブ上の数百万のウェブサイトと大量のページ読み込みを対象としています。データ収集と方法論の詳細は方法論で確認できます。

267. *https://web.dev/articles/user-centric-performance-metrics*
268. *https://web.dev/articles/lcp*
269. *https://web.dev/articles/inp*
270. *https://web.dev/articles/cls*
271. *https://www.debugbear.com/blog/firefox-safari-web-vitals*
272. *https://web.dev/articles/ttfb*
273. *https://web.dev/articles/fcp*
274. *https://web.dev/performance*
275. *https://developer.chrome.com/docs/web-platform/early-hints*
276. *https://developer.chrome.com/docs/web-platform/implementing-speculation-rules*
277. *https://httparchive.org/faq*
278. *https://developer.chrome.com/docs/crux*

Core Web Vitalsの概要

Core Web VitalsはGoogleが実際のユーザーにウェブページがどのように感じられるかを理解するための主要な指標です。以下の条件を満たすページが「良好」と見なされます：

- **Largest Contentful Paint（LCP）**：メインコンテンツが素早く表示される（2.5秒以内）ため、ページが有用に感じられる。
- **Interaction to Next Paint（INP）**：クリックやタップへのページの反応がほぼ即時（200ミリ秒以内）。
- **Cumulative Layout Shift（CLS）**：レイアウトがほぼ安定しており、予期しない移動がほとんどない（スコア ≤ 0.1）。

ほとんどのユーザーに対してこれらのしきい値を満たすページは、「良好」な全体的なページ体験を提供していると分類されます。

もちろん、これらはウェブ全体の推定的な分類を目的とした大まかな指標です。個々のウェブサイトでは、ユーザーにとっての「良好」の期待値が異なる場合があります。

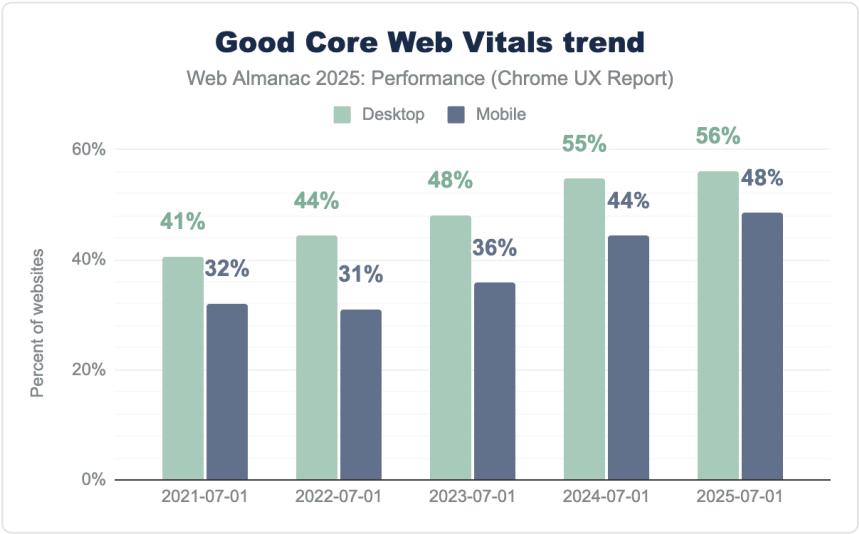


図7.1. 良好なCore Web Vitalsのトレンド。

モバイルのCore Web Vitalsは継続的な前年比改善を示しており、2023年の36%から2024年の44%に増加し、2025年には48%に達しました。この上昇はサイトの最適化に加え、ブラウザ、デバイス、ネットワークの改善を反映している可能性があります。

デスクトップのパフォーマンスも、2023年の48%から2024年の55%へとポジティブなトレンドを示しました。ただし、2025年の改善は僅かで、56%への増加にとどまりました。

これらのトレンドをより深く理解するために、次のセクションではCore Web Vitalsがページの人気度によってどのように異なるかを調べます。より人気のあるページはランク値が低く表示されます。

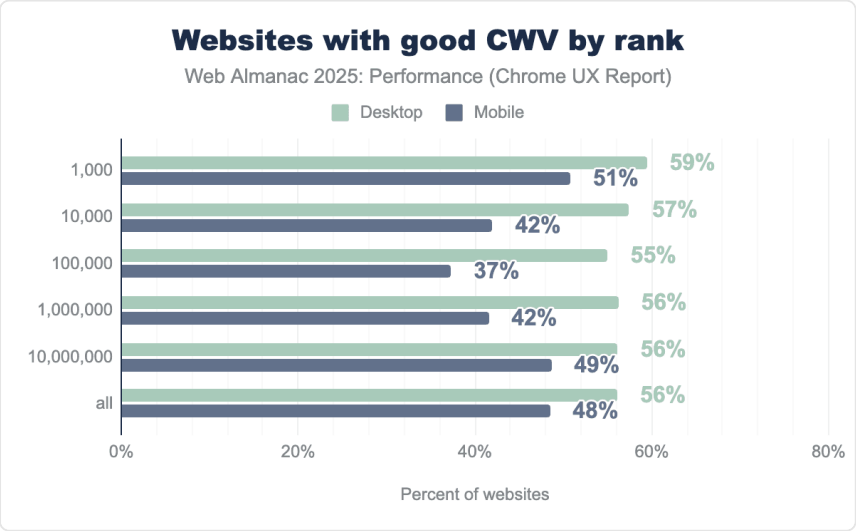


図7.2. ランク別の良好なCWVを持つウェブサイト。

モバイルでは、最も人気のあるサイトと最も人気のないサイトが、人気分布の中間に位置するサイトよりも優れたパフォーマンスを示す傾向があります。最も人気のあるサイトはCore Web Vitalsの結果が良好な一方、中程度の人気のサイトではパフォーマンスが低下し、最も人気のないサイトでは再び改善されます。

- 最も人気のある1,000のモバイルサイトの51%が良好なCore Web Vitals（CWV）を持つ。
- CWVは次の10,000サイトで42%、次の100,000サイトで37%に低下する。
- しかし、次の1,000,000サイトで42%、次の10,000,000サイトで48%に改善する。

このパターンは、人気ティア間のページの複雑さとパフォーマンスへの投資の違いを反映している可能性があります。

- 人気の高いサイトはパフォーマンスを優先事項として扱い、ユーザーエンゲージ

メントとビジネス成果との密接な相関関係[279]から継続的な最適化に投資する可能性が高い。

- 中程度の人気のサイトは、追加機能やサードパーティスクリプトなど高い複雑さとパフォーマンスへの継続的な焦点の欠如を組み合わせ、結果の低下につながる場合がある。
- 人気の低いサイトは多くの場合よりシンプルで、機能が少なく軽量なページを持ち、プラットフォームのデフォルト設定の恩恵を受けて比較的優れたパフォーマンスを提供できる。

このU字型のパターンはモバイルでより顕著で、より遅いデバイスと不安定なネットワーク状況がページの複雑さと最適化の限界の影響を増幅させる傾向があります。デスクトップでは、より強力なハードウェアと安定したネットワークにより、これらの違いの目に見える影響を軽減できます。

パフォーマンスはプライマリナビゲーションとセカンダリナビゲーション間でも大きく異なる場合があります。プライマリナビゲーションは通常、ユーザーがホームページのサイトにアクセスするときに発生し、より多くのリソースをフェッチして実行する必要があります。一方、セカンダリナビゲーションは、ユーザーが同じサイト内のページ間を移動する際に発生し、以前に読み込まれてキャッシュされたリソースの恩恵を受けることができます。

この章のほとんどのCrUXデータはオリジン単位ですが、クローラーはクロール時に利用可能な場合はページレベルのCrUXデータも収集しており、ホームページとセカンダリページナビゲーション間のCore Web Vitalsの違いを調べることができます。

279. *https://www.speedcurve.com/blog/site-speed-business-correlation/*

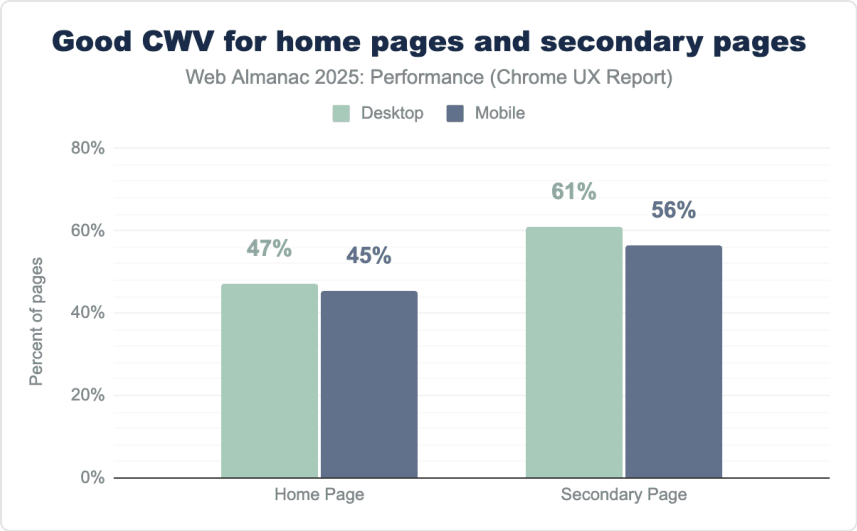


図7.3. ホームページとセカンダリページの良好なCWV。

セカンダリページはホームページよりも高いCWV合格率を示しており、デスクトップで14%、モバイルで11%のリードがあります。このパフォーマンスの差は、セカンダリページがキャッシュされた情報の恩恵を受けることが多く、それがより速いページ読み込みに貢献していることを示唆しています。ホームページはより頻繁に更新され、より動的でさまざまなコンポーネントを含む傾向がある一方、セカンダリページはよりテンプレート化されて一貫性があり、より安定して最適化しやすい場合があります。

現代のシングルページウェブサイトは、フルページリロードなしにコンテンツが変わるJavaScriptベースのナビゲーションを使用することが多いです。これらのナビゲーションはユーザーにとってページ間の移動のように感じられますが、現在のWeb Vitalsの計測では常に完全に捕捉されるわけではありません。ソフトナビゲーション[280]のサポートにより、これらのページ内遷移でのCore Web Vitalsの捕捉が改善され、最初のページ読み込みを超えた実際のユーザー体験のより正確なビューが提供されることが期待されています。

読み込み速度

ユーザーの品質と信頼性の認識に影響を与える主要な要因の1つが、ウェブサイトの初期読み込み速度です。しかし、「速度」はウェブサイトの文脈において本質的に相対的であり、単一の値で定義するのは難しいです。パフォーマンスはユーザーのデバイス能力とネットワーク環境によって異なるため、ユーザー体験を捉えるために単一の「読み込み時間」に頼る

280. *https://developer.chrome.com/blog/new-soft-navigations-origin-trial*

ことはできません。そのため、サイトがどれだけ速く読み込まれるかだけでなく、どれだけ速く *感じられる* かを計測する複数のユーザー中心の指標[281]を見ていきます。

以下のセクションでは、First Contentful Paint（FCP）とLargest Contentful Paint（LCP）の2つの主要な読み込み指標に焦点を当てます。

First Contentful Paint

ウェブページの速度に対するユーザーの第一印象を理解するために、First Contentful Paint（FCP）[282]を見ていきます。この指標は、ユーザーが最初にページをリクエストした時点から計測して、ページが *あらゆる* コンテンツを表示し始めるまでにかかる正確な時間を捉えます。FCPスコアが1.8秒未満のページは「良好」と見なされ、1.8〜3.0秒のスコアはページが「改善が必要」であることを示し、3.0秒を超えるスコアは「不良」なパフォーマンスと見なされます。

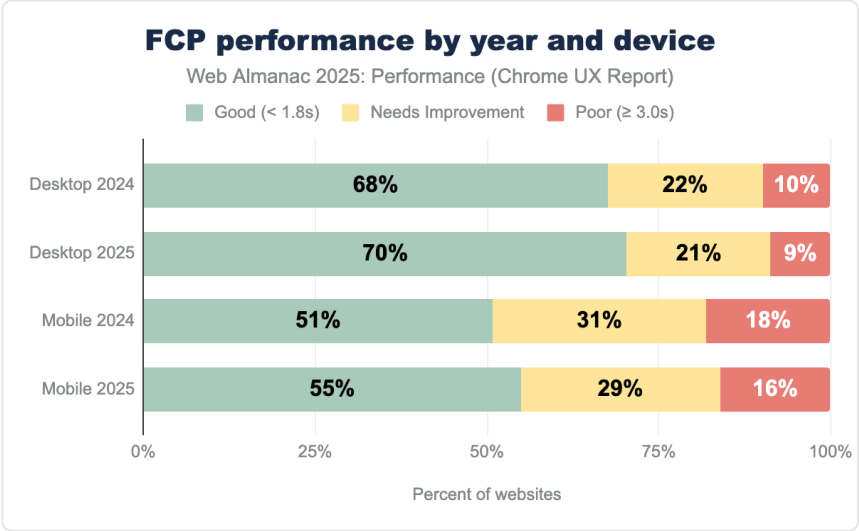


図7.4. 年別・デバイス別のFCPパフォーマンス。

FCPパフォーマンスは2024年以降、デスクトップとモバイルの両方で改善されました。「良好」なFCPを達成するデスクトップサイトの割合が2%増加し、モバイルサイトはより大きな4%の改善を見せました。FCPは大まかに2つの主要な部分から構成されていると理解でき、それぞれが読み込みプロセスの異なる側面によって影響を受けます：

281. *https://web.dev/articles/user-centric-performance-metrics*
282. *https://web.dev/articles/fcp*

- 1つ目は、Time to First Byte（TTFB）[283]で捉えられるネットワークとサーバーのオーバーヘッドです。これには接続セットアップ、リダイレクト、サーバー処理時間が含まれ、主にネットワークインフラとプロトコル効率によって影響を受けます。Service Workerがキャッシュからレスポンスを提供する場合、ネットワークのラウンドトリップを回避でき、再訪問時のTTFBを改善できます。ただし、Service Workerの起動も遅延を加える可能性があり、Navigation Preload[284]は初期化中にネットワークリクエストを並行して開始することでこれを軽減するのに役立ちます。

- 2つ目はクライアントサイドレンダリングで、最初のバイトを受信した後に始まります。これはブラウザがリソースを解析してページの最初の可視コンテンツをレンダリングするために必要な時間を反映しており、ブラウザの動作、レンダリングブロックリソース、ユーザーハードウェア能力などの要因によって影響を受けます。

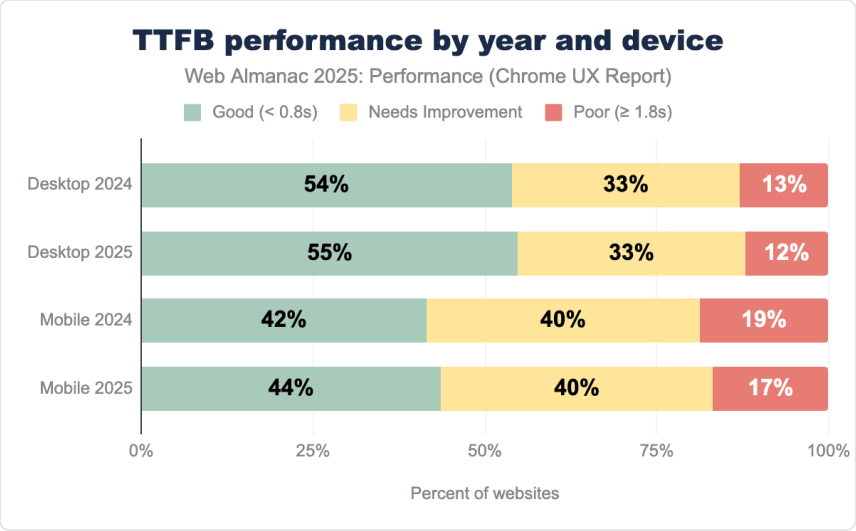


図7.5. 年別・デバイス別のTTFBパフォーマンス。

2024年以降、「良好」なTTFBを達成するサイトの割合はデスクトップで1%、モバイルで2%増加しました。

283. https://web.dev/articles/ttfb
284. https://web.dev/blog/navigation-preload

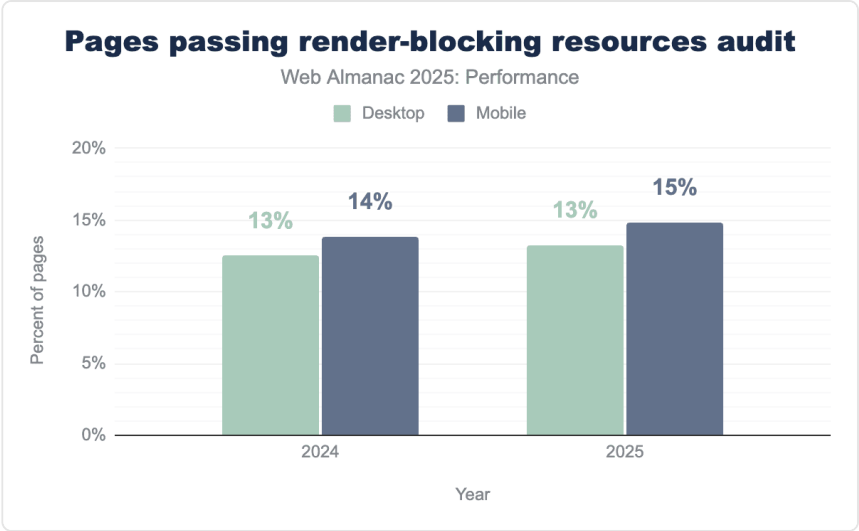


図7.6. レンダリングブロックLighthouse監査に合格したページ。

同期間、「レンダリングブロックリソース監査」に合格したページの割合はデスクトップで横ばい、モバイルで1%増加しました。

まとめると、2024年から2025年にかけてのFCPの改善は、サーバーレスポンスタイムのわずかな改善とレンダリングブロック作業のわずかな削減と一致しています。これは、ネットワーク配信とクライアントサイドレンダリングの両方での段階的な改善が、より早い最初の描画に貢献しており、モバイルデバイスへの影響がわずかに大きいことを示唆しています。

FCPが最初の視覚的反応を捉えるのに対し、LCPはページの主要なコンテンツがいつ表示されるかを反映し、通常より長く複雑なクリティカルパスを伴います。FCPと同様に、LCPはいくつかの連続したフェーズの合計として理解できます：サーバーから最初のバイトを受信するまでの時間（TTFB）、ブラウザがLCPリソースのフェッチを開始するまでの遅延（リソース読み込み遅延）、そのリソースの読み込みに費やされる時間（リソース読み込み時間）、および要素がレンダリングされる前の遅延（要素レンダリング遅延）。これらのフェーズ全体で時間がどこに費やされているかを理解することが、LCP、ひいては全体的なCore Web Vitalsパフォーマンス[285]の改善に重要です。

Largest Contentful Paint

ページがいつ意味のある形で読み込まれたと感じるかを理解するために、Largest Contentful

285. *https://web.dev/articles/defining-core-web-vitals-thresholds*

Paint（LCP）[286]を見ていきます。この指標は、ユーザーが最初にページをリクエストした時点から、最大の可視要素（通常はヒーロー画像、見出し、または目立つテキストブロック）が画面上でレンダリングを終了するまでの時間を計測します。LCPスコアが2.5秒未満のページは「良好」と見なされ、2.5〜4.0秒のスコアはページが「改善が必要」であることを示し、4.0秒を超えるスコアは「不良」なパフォーマンスと見なされます。

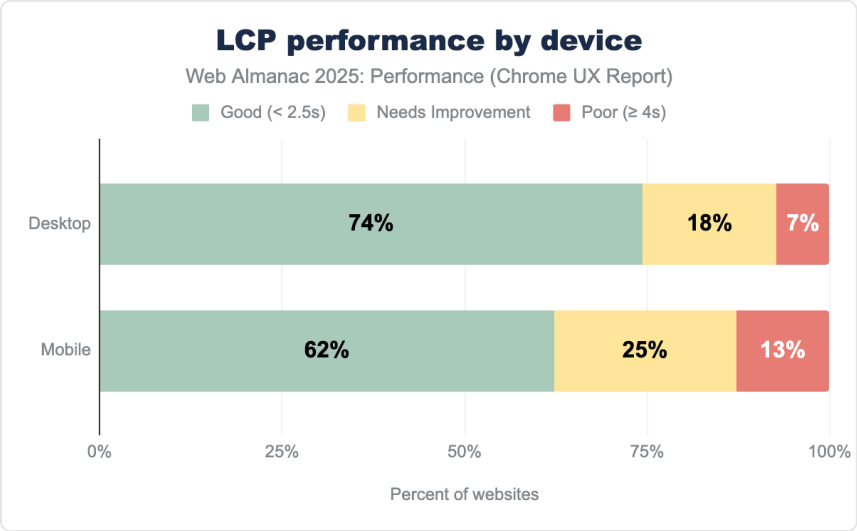


図7.7. デバイス別のLCPパフォーマンス。

現在、デスクトップページの74%が「良好」なLCPスコアを達成しているのに対し、モバイルは62%で、モバイルは「不良」な体験の率もほぼ2倍（13%対7%）を示しています。このギャップは、モバイルにおけるより遅いネットワークとより低性能なデバイスの複合的な影響と一致しています。

LCPコンテンツタイプ

LCPを効果的に最適化するには、まずどのタイプのコンテンツが通常LCP要素になるかを理解する必要があります。

286. https://web.dev/articles/lcp

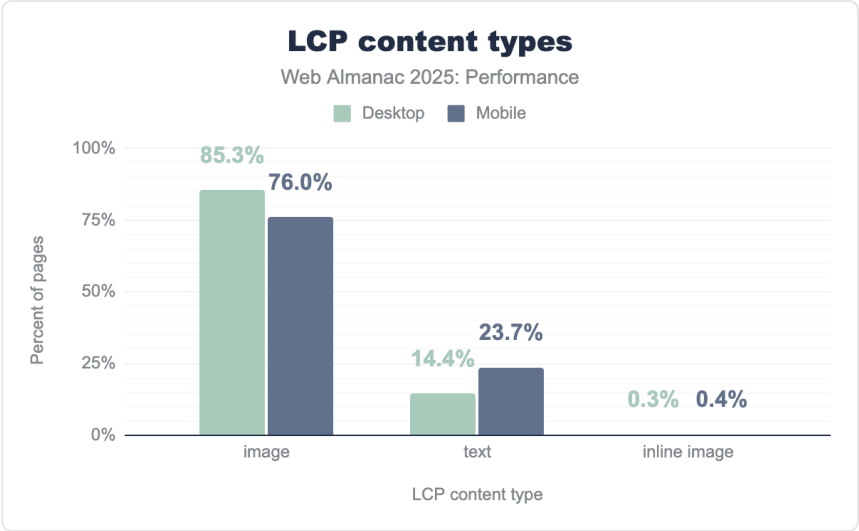


図7.8. LCPコンテンツタイプ。

LCPコンテンツタイプのトレンドは過去の年と同様です（2022年[287]および2024年[288]のデータも参照）。画像は両デバイスタイプのLCP要素として引き続き支配的で、デスクトップページの85.3%とモバイルページの76%が画像をLCP要素として持っています。テキストベースのLCP要素が残りの多くを占めており、デスクトップで14.4%、モバイルで23.7%です。このギャップは、狭いビューポートでヒーロー画像がリサイズされ、より小さなビジュアルに置き換えられるか、または完全に削除されるレスポンシブデザインの慣行を反映している可能性が高く、代わりに見出しテキストが最大の可視要素になります。

インライン画像（HTMLに直接埋め込まれたデータURI）はページの約0.5%でまれなままで、より大きなHTMLペイロードとキャッシュ効率に関するトレードオフについての限定的で慎重な採用と意識を示しています。

LCP画像フォーマット

LCP要素としての画像のこの継続的な支配を考慮すると、LCPのリソース読み込み時間フェーズに直接影響するため、使用されている画像フォーマットを見ることが重要になります。2024年のチャプターではこのフェーズが他と比べて最適化の可能性が低いことが示されましたが[289]、画像フォーマットの効率性は依然として全体的なパフォーマンスに貢献します。

287. *https://almanac.httparchive.org/ja/2022/performance#fig-8*
288. *https://almanac.httparchive.org/ja/2024/performance#lcpコンテンツタイプ*
289. *https://almanac.httparchive.org/ja/2024/performance#lcpサブパーツ*

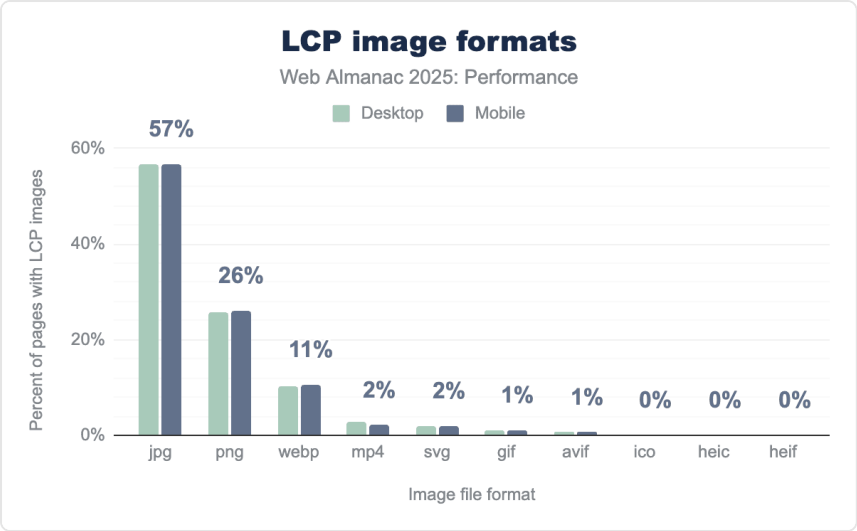


図7.9. LCP画像フォーマット。

WebPやAVIFなどの最新フォーマットはレガシーフォーマットよりも優れた圧縮を提供し、ファイルサイズが小さく転送が速くなります。しかし、レガシーのJPGとPNGは依然として多く使用されています（JPGがLCP画像の57%、PNGが26%を占めています）。

ただし、JPGの使用が2024年以降[290]4%減少し、WebPが4%増加しているなど、いくつかの前向きな兆候があります。

PNGやその他のフォーマットが2024年の割合と同じである（AVIFが0.7%に達したことを除いて）ことから、ウェブページはゆっくりとではありますが、JPGからWebPに移行しているようです。この遅い採用は、最新フォーマットが広いサポートを持っているにもかかわらず、既存の画像パイプラインとコンテンツライブラリを移行するコストを反映しているかもしれません。

クロスオリジンLCP画像

LCP画像のオリジンは、ブラウザがダウンロードを開始できる速さに影響し、リソース読み込み遅延フェーズに影響します。画像がページと同じドメインでホストされている場合、ブラウザは既存の接続を再利用できます。クロスオリジン画像は、特にオリジンがまだ接続されていない場合、追加の接続セットアップ（DNS/TCP/TLS）を生じさせる可能性があり、ダウンロードが開始できるまでの時間が増加します。

290. *https://almanac.httparchive.org/ja/2024/performance#fig-19*

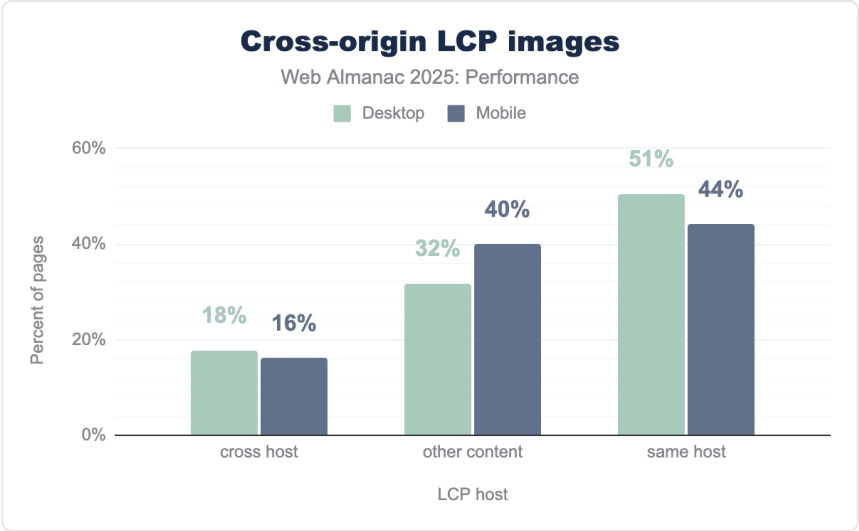


図7.10. クロスオリジンLCP画像。

デスクトップページの51%とモバイルページの44%は、ドキュメントと同じホストからLCP画像を提供しています。クロスホストのLCP画像はページの16〜18%を占めており、preconnectヒント[291]で軽減されない限り、接続オーバーヘッドのコストを支払っている可能性がある意味のある割合です。

「その他のコンテンツ」カテゴリ（デスクトップ32%、モバイル40%）は、LCP要素が画像でないページ、おそらくテキストブロックや背景要素を表しています。「その他のコンテンツ」のモバイルの割合が高いのは、狭いビューポートでヒーロー画像が優先度を下げられるレスポンシブデザインパターンを反映しているかもしれませんが、このデータだけでは確定的にはわかりません。

LCPリソースの優先度設定

リソース読み込み遅延フェーズはLCP時間の大部分を占めることが多いため、ブラウザはクリティカルリソースを加速するためのツールを提供しています。`fetchpriority="high"`属性は、ブラウザが通常よりも高い優先度でリソースを優先するよう指示します。これは画像がLCP要素であっても通常は高優先度と見なされないため有用です。一方、`<link rel="preload">`はブラウザがHTMLで自然に発見する前にリソースをフェッチするよう指示します。

291. *https://web.dev/learn/performance/resource-hints#preconnect*

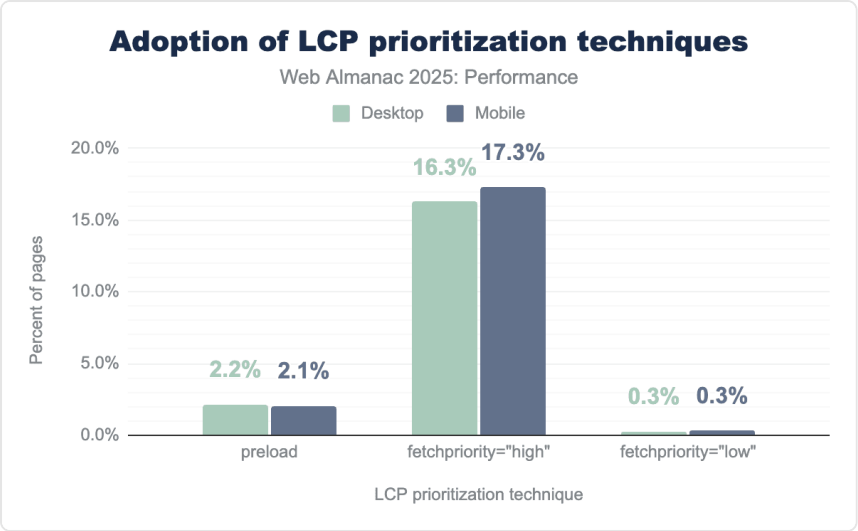


図7.11. LCP優先度設定技術の採用。

`fetchpriority="high"` の採用は成長を続けており、現在LCP画像を持つモバイルページの17%に登場しています（2024年の15%[292]から増加）。preloadの使用は2.1〜2.2%と低いままです。

両技術は比較的実装が簡単ですが、preloadはリソースがHTMLドキュメントで素早く発見できない場合にのみ必要であることに注意すべきです。また、preloadを使用する場合は、`fetchpriority="high"` と組み合わせて画像が高優先度でフェッチされることを確保すべきです（preloadだけを使用しても保証されません）。

LCP画像に `fetchpriority="low"` を使用しているページの0.3%は意図的でない可能性があります。開発時にどの画像がLCP要素になるかを特定することは、開発者にとってトリッキーな場合があるためです（ビューポートとコンテンツによって異なります）。

LCP遅延読み込み

画像の遅延読み込みとは、画像が必要になるまで読み込みを遅らせることを意味します。例えば、ページ読み込み時にすべての画像を読み込むデフォルトの代わりに、フォールドより下の画像をユーザーのビューポートに近づいたときにのみ読み込みます。これは重要なフォールドより上のコンテンツを優先するのに役立ちます。遅延読み込みは一般的に有用な最適化ですが、LCP画像に適用するとユーザーが待っているメインコンテンツの表示が遅れるため、有害になる可能性があります。

292. *https://almanac.httparchive.org/ja/2024/performance#lcp優先順位付け*

図7.12. LCP画像を遅延読み込みしているモバイルページの割合。

全体として、約16〜17%のページがLCP画像を遅延読み込みしており、この数字は2024年以降変わらず安定しています。ただし、構成は変化しています：ネイティブの `loading="lazy"` の使用がわずかに増加し（モバイルで9.5%から10.4%、デスクトップで10.2%から11.5%）、`data-src` 属性の背後にソースを隠すカスタムアプローチは減少しています（モバイルで6.7%から5.9%）。ネイティブの `loading="lazy"` はカスタムアプローチよりもLCP遅延読み込みの大きなシェアを占めており、標準化されたブラウザ機能への移行を示しています。

読み込み速度の結論

まとめると、読み込み指標は以下の主要なトレンドを浮き彫りにしています：

- FCPとLCPはともに2024年以降改善されており、デスクトップは一貫してモバイルを上回っています。
- 新しい画像フォーマットの採用は、JPGからWebPへの緩やかな移行にもかかわらず、依然として限られています。
- 約16%のウェブページがまだLCP画像を遅延読み込みしており、プライマリコンテンツの表示を遅らせています。

インタラクティビティ

歴史的にウェブパフォーマンスの計測は読み込み速度に集中してきましたが、INPのような指標のおかげで、ページが読み込まれた後のインタラクティビティを計測することが同等、あるいはそれ以上に重要であるという認識が高まっています。

Interaction to Next Paint（INP）

Interaction to Next Paint（INP）[293]は、セッション中にページと行われたすべてのインタラクションを観察し、最悪の遅延を報告することで計算されます（ほとんどのサイトでは、多くのインタラクションを持つページが外れ値を無視するための追加の余裕があります）。イン

293. https://web.dev/articles/inp

タラクションの遅延は、ユーザーがインタラクションを開始した時点から、ブラウザが次にフレームを描画できる瞬間まで、インタラクションを駆動するイベントハンドラーのグループの単一の最長期間で構成されます。

オリジンが「良好」なINPスコアを受け取るには、すべてのセッションの少なくとも75%が200ミリ秒以下のINPスコアを必要とします。INPスコアは、ページ上のすべてのインタラクションの中で最も遅い、またはほぼ最も遅いインタラクション時間です。詳細については、INPの計算方法の詳細[294]を参照してください。

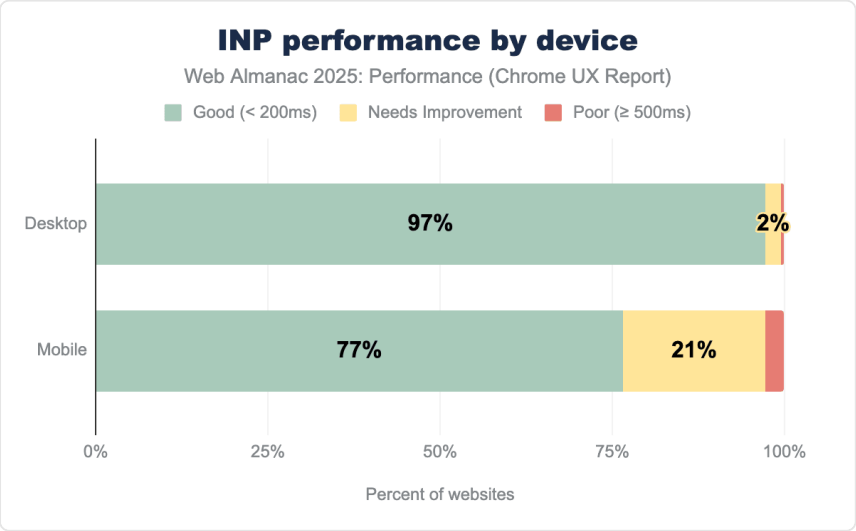


図7.13. デバイス別のINPパフォーマンス。

2025年、モバイルのINPパフォーマンスは前向きな改善を示し、77%のウェブサイトが良好なスコアを達成しました（2024年の74%[295]から上昇）。この3パーセントポイントの向上は、何百万ものウェブサイトが今やモバイルユーザーにより応答性の高い体験を提供していることから、意味のある進歩を表しています。デスクトップのパフォーマンスは97%と引き続き模範的で、過去数年に確立された高い水準を維持しています。

注目すべきことに、モバイルとデスクトップのパフォーマンスギャップは縮小し始めており、2024年の23パーセントポイントから2025年の20パーセントポイントに縮小しました。20パーセントポイントのギャップはまだ大きいですが、これはギャップを縮める方向への最初の測定可能な一歩です。このトレンドは、モバイル最適化の取り組みがウェブ全体で勢いを増していることを示しています。

294. *https://web.dev/articles/inp#good-score*
295. *https://almanac.httparchive.org/ja/2024/performance#interaction-to-next-paint-inp*

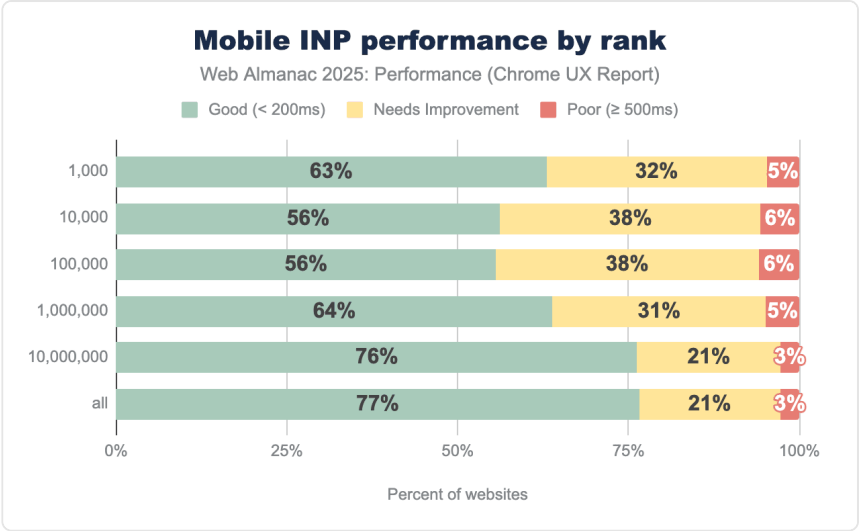


図7.14. ランク別のモバイルデバイスのINPパフォーマンス。

最も人気のあるウェブサイトは2025年に目覚ましいINPの改善を示し、上位1,000サイトは良好なスコアが53%から63%に急増しました。これは10パーセントポイントの向上で、他のすべてのカテゴリを上回りました。これは、高トラフィックのウェブサイトがインタラクティビティの最適化を優先していることを示しており、ユーザーエンゲージメントとビジネス指標への直接的な影響によって促進されている可能性が高いです。

人気サイトは全体平均の77%をまだ下回っていますが、ギャップは大幅に縮まりました。上位1,000サイトは2024年に平均より21パーセントポイント下でしたが、2025年には14パーセントポイント下にとどまっており、どのカテゴリでも観察された最速の改善率です。

このパターンは、高トラフィックウェブサイトが直面する独自の課題を反映しています：より複雑な機能、より豊かなインタラクティブ機能、より重いサードパーティ統合、および多様なユーザーインタラクションパターン。Eコマースプラットフォーム、ソーシャルメディアサイト、ニュースポータルは本質的により多くのJavaScript実行を必要とし、良好なINPスコアの達成をより困難にしています。

大幅な前年比改善は、主要なウェブサイトがコード分割、インタラクション最適化、選択的機能読み込みを通じてこれらの課題を成功裏に解決していることを示唆しています。最も訪問されるサイトがパフォーマンスを向上させ続けるにつれて、より高い基準を設定し、より広いウェブエコシステムに価値ある最適化パターンを提供しています。

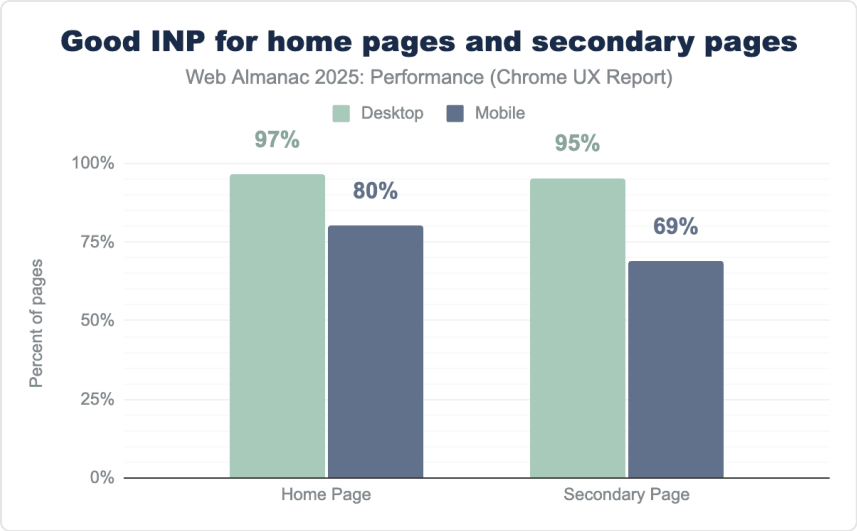


図7.15. ホームページとセカンダリページの良好なINP。

ページレベルのCrUXデータを見ると、2024年から[296]の注目すべき変化として、ホームページが今やモバイルデバイスでセカンダリページよりも大幅に良好なINPパフォーマンスを示しています。モバイルホームページは80%の良好なINPスコアを達成し、2024年より7パーセントポイント改善されました。セカンダリページは69%に低下し、11パーセントポイントのギャップを生じさせました。この乖離は、ホームとセカンダリページがほぼ同一のパフォーマンスを示した2024年（モバイルで73%対72%）からの変化を表しています。デスクトップのパフォーマンスは両ページタイプでそれぞれ97%と95%と堅調を維持しました。

ホームページのINPの改善は、第一印象が重要なランディングページへの最適化の注力が増加したことを反映している可能性があります。しかし、セカンダリページのパフォーマンス低下は注意が必要で、これらのページはフィルター、カルーセル、フォーム検証などのより複雑なインタラクションを含むことが多く、またユーザージャーニーの深いところで活性化するサードパーティウィジェットと分析からのJavaScriptも蓄積するためです。

Total Blocking Time（TBT）

Total Blocking Time（TBT）[297]は、First Contentful Paint（FCP）後にメインスレッドが入力応答性を防ぐほど長くブロックされた合計時間を計測します。

TBTはラボ指標であり、実ユーザーモニタリングを使用してのみ収集できるINPのようなフ

296. https://almanac.httparchive.org/ja/2024/performance#fig-21
297. https://web.dev/articles/tbt

ィールドベースの応答性指標のプロキシとしてよく使用されます。ラボベースのTBTとフィールドベースのINPは相関しており[298]、TBTの結果は一般的にINPのトレンドを反映しています。200ミリ秒未満のTBTは良好と見なされますが、ほとんどのモバイルウェブサイトはこの目標を大幅に超えています。

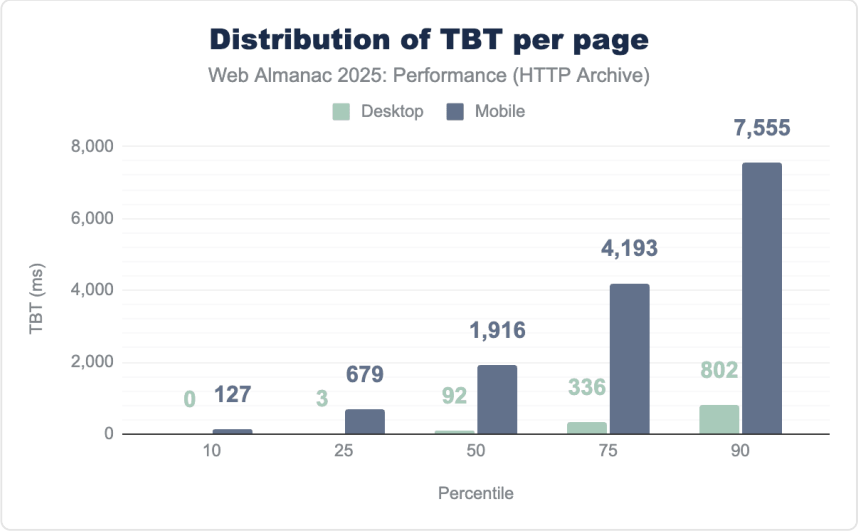


図7.16. ページあたりのTBTの分布。

モバイルの中央値TBTは2025年に1,916ミリ秒に増加し、2024年の1,209ミリ秒から58%上昇[299]しました。デスクトップのTBTも67ミリ秒から92ミリ秒に上昇しました。90パーセンタイルでは、モバイルユーザーはページが完全にインタラクティブになる前に7.5秒以上のブロック時間に直面しています。

これは明らかな矛盾を提示しています：フィールドベースのINPスコアが改善されたにもかかわらず、ラボベースのTBTは大幅に悪化しました。この乖離の背後にはいくつかの要因が考えられます。

- 現実世界のデバイスがより強力になり、ラボテストが一貫したエミュレートデバイスを使用して明らかにするコードの複雑さの増加を隠しています。
- 一部のサイトはINPを支配するインタラクションを最適化しながら、TBTに表れる大量のバックグラウンド作業を実行し続けている可能性があります。
- INP指標は進化し続けており、ChromiumのINP指標の変更履歴[300]に記載されてい

298. https://colab.research.google.com/drive/12lJmAABgyVjaUbmWvrbzj9BkkTxw6ay2
299. https://almanac.httparchive.org/ja/2024/performance#total-blocking-time-tbt
300. https://chromium.googlesource.com/chromium/src/+/main/docs/speed/metrics_changelog/inp.md

るように、計測の安定化と現実世界のインタラクション動作のより良い捕捉に焦点を当てた今後の改善が予定されています。

デスクトップ（中央値92ms）とモバイル（中央値1,916ms）のギャップの拡大は、デバイスクラス間の持続的なパフォーマンスの不平等を強調しており、INPの改善にもかかわらず、メインスレッドブロッキングの根本的な課題が激化していることを示唆しています。

インタラクティビティの結論

インタラクティビティの結果の主なポイントは以下の通りです：

- モバイルINPは77%に改善し（74%から上昇）、モバイルとデスクトップのギャップを20パーセントポイントに縮小しました。
- 上位1,000のウェブサイトが最も大きな改善を達成し、良好なINPが53%から63%に向上しました。
- ホームページはセカンダリページを大幅に上回るようになりました（モバイルで80%対69%）。
- INPの改善にもかかわらず、TBTは58%増加し、全体的なJavaScript実行の重さが増していることを示しています。

視覚的安定性

視覚的安定性はCumulative Layout Shift（CLS）によって計測され、ページがユーザーにとってどれだけ予測可能でスムーズに感じられるかの指標です。このセクションでは、最近の数年間の進歩、デバイスの違い、および変化に焦点を当てます。

Cumulative Layout Shift（CLS）

Cumulative Layout Shift（CLS）は、ページの読み込みとインタラクション中の予期しないレイアウトの動きを計測し、スコアが高いほど視覚的なシフトが多くなります。CLSスコアは3つのしきい値に分類されます：「良好」（≤ 0.1）、「改善が必要」（> 0.1かつ ≤ 0.25）、「不良」（> 0.25）。これにより、ウェブサイト間の視覚的安定性を評価・比較するための標準化された方法が提供されます。

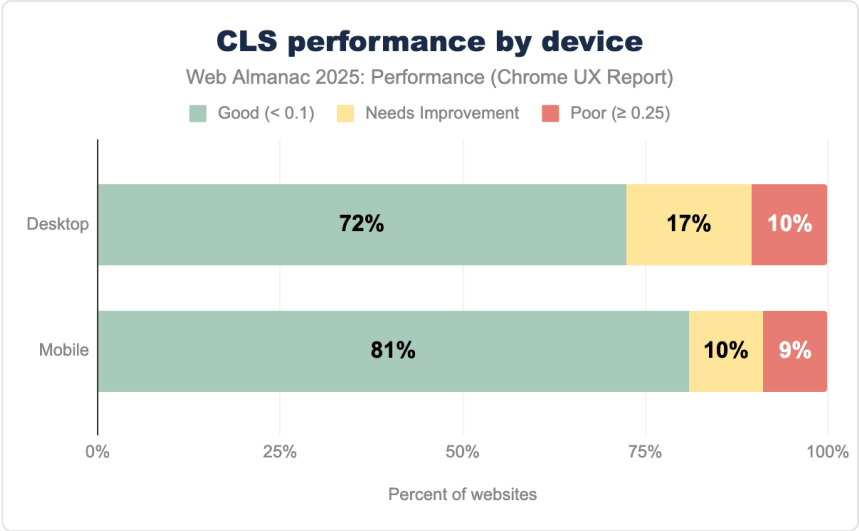


図7.17. デバイス別のCLSパフォーマンス

2025年、デスクトップページの72%とモバイルページの81%が「良好」なCumulative Layout Shift（CLS）スコアを達成しています。デスクトップページはモバイル（10%）に比べてCLSが「改善が必要」な割合が高く（17%）、「不良」なCLSのページの割合はデバイス間でほぼ同じで約9〜10%です。これは、ほとんどのページがCLSのしきい値を満たすのに近く、深刻なレイアウト不安定性を経験するページが少ないことを示しています。

2024年と比較して[301]、「不良」なCLSを持つデスクトップページの割合が1%減少し、「改善が必要」と分類されるページの割合が同様に増加しました。

301. *https://almanac.httparchive.org/ja/2024/performance#cumulative-layout-shift-cls*

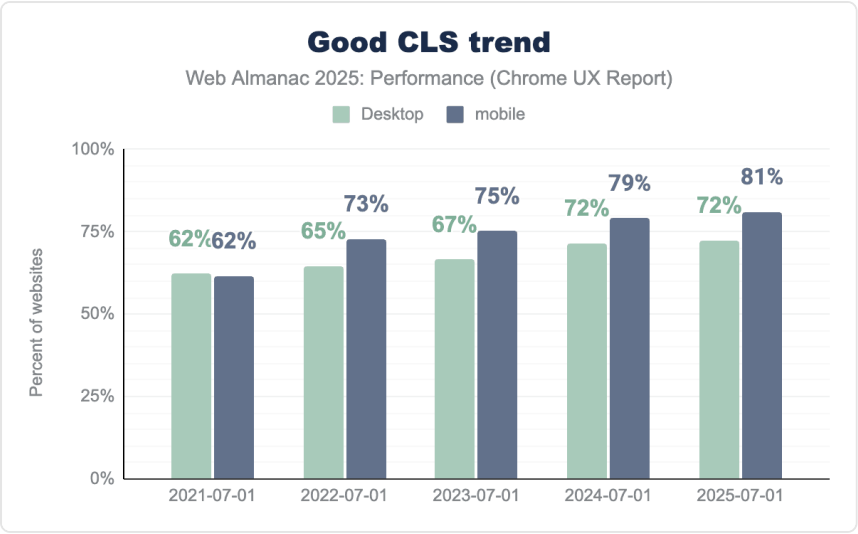


図7.18. 2021年から2025年のデバイス別CLSパフォーマンス

過去の年を振り返ると、「良好」なCLSのしきい値を満たすウェブサイトの割合はデスクトップとモバイルの両方で毎年増加しています。デスクトップのCLSは2021年の62%から2025年の72%へと徐々に改善し、モバイルは同期間に81%と、より大きな改善を見せました。ただし、昨年と比較した増加はわずかで、デスクトップで「良好」なCLSのしきい値を満たすサイトの割合は変わらず、モバイルは2%しか改善されていません。

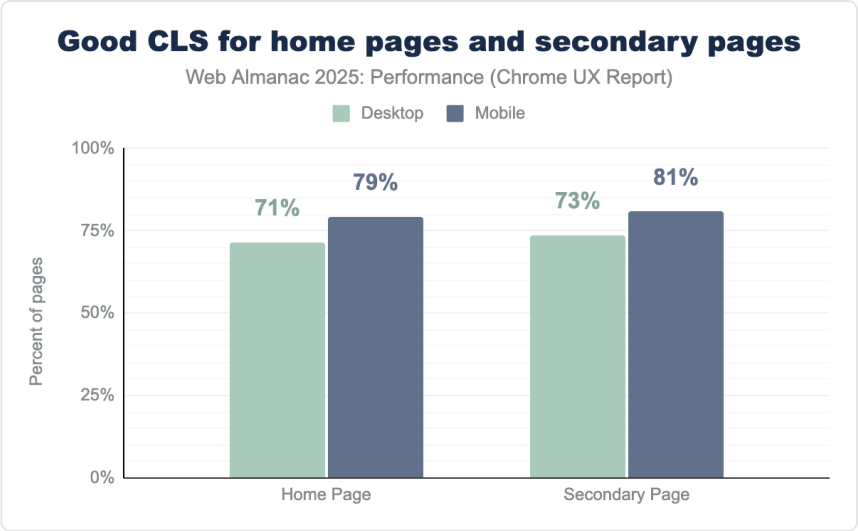


図7.19. ページタイプ別の良好なCWVを持つウェブサイトの割合。

ページレベルのCrUXデータを見ると、ホームページ以外のページはデスクトップとモバイルの両デバイスでホームページよりもわずかに良好な視覚的安定性を示しています。2025年、デスクトップのセカンダリページの73%が「良好」なCLSを達成しており、デスクトップホームページの71%と比較されます。モバイルでは、セカンダリページの81%が「良好」なCLSのしきい値を満たしており、モバイルホームページの79%と比較されます。これは、ヒーローメディア、バナー、プロモーション要素などのより動的なコンテンツを含むことが多いホームページが、セカンダリページよりもレイアウトシフトを起こしやすいことを示唆しています。

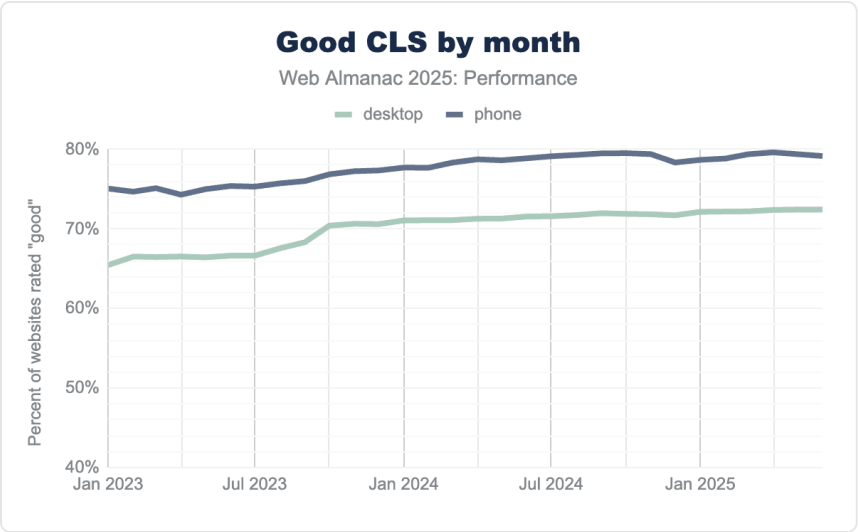


図7.20. 2023年から2025年のデバイス別月次CLSトレンド。

2023年から2025年にかけて、「良好」なCLSを持つサイトの割合は両デバイスタイプで着実に増加しており、モバイルが一貫してデスクトップを上回っています。時間の経過とともに若干の変動がありますが、両トレンドとも急激な変曲点なしに緩やかな上昇軌跡を示しており、突然の変化ではなく継続的な改善を示しています。

CLSのベストプラクティス

サイトがCLSの可能性を低減するために従えるベストプラクティスが多数あります。

バック/フォワードキャッシュ（bfcache）

バック/フォワードキャッシュ（bfcache）[302]は、ユーザーがブラウザの戻るまたは進むボタンを使用してナビゲートするときに、ブラウザがページをメモリから即座に復元できるようにします。ページをリロードしてJavaScriptを再実行するのではなく、ブラウザはページの状態を保持し、ほぼ瞬時のナビゲーションと改善されたユーザー体験をもたらします。ページが以前の状態に復元されるため、bfcacheは再ナビゲーション中に発生する可能性のあるレイアウトシフトを回避するのにも役立ちます。

ただし、すべてのページがbfcacheの対象ではありません。対象資格はページライフサイクル要件のセットに依存しており、これらの制約に違反するページは完全なリロードにフォー

302. *https://web.dev/articles/bfcache*

ルバックします。対象外の理由のリストはHTML仕様[303]で確認できます。bfcacheの動作は最終的にブラウザによって処理されますが、開発者はChrome DevToolsを使用してページの適格性を評価[304]できます。

ページは既知のライフサイクルの動作によってbfcacheから除外される場合があります。これには、unloadまたは `beforeunload` イベントハンドラーの使用、アクティブな接続や管理されていないタイマーなどの復元できない副作用、および安全なページの復元に干渉する特定のサードパーティスクリプトが含まれます。そのため、unloadイベントはパフォーマンスへの負の影響とバック/フォワードキャッシュとの非互換性のために非推奨とされ、推奨されていません。

Chromeチームは、最近の使用パターンに反映されているように、`visibilitychange` や `pagehide` などの代替手段を支持して `unload` の回避を推奨しています。2024年と比較して[305]、2025年にはすべてのランクと両デバイスでアンロードハンドラーの使用が減少しました。この削減は、より多くのページがbfcacheの動作の対象になったことを示唆しています。この進歩にもかかわらず、アンロードハンドラーは依然として上位ランクのサイトとデスクトップでより一般的で、以下のグラフに示すように、ウェブのかなりの部分のbfcacheの適格性を制限し続けています。

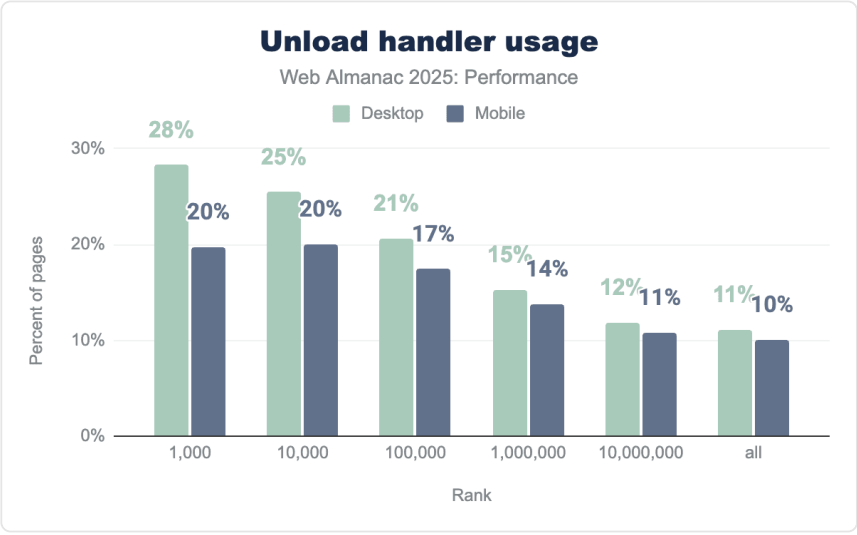


図7.21. ウェブサイトランクとデバイス別のアンロードハンドラー使用状況（2025年）

サイトの人気が下がるにつれてアンロードハンドラーの使用が一貫して減少するのは興味深いです。高トラフィックのウェブサイト（上位1,000サイト）では、アンロードハンドラー

303. https://html.spec.whatwg.org/multipage/nav-history-apis.html#nrr-details-reason
304. https://developer.chrome.com/docs/devtools/application/back-forward-cache
305. https://almanac.httparchive.org/ja/2024/performance#backforward-cache-bfcache

がデスクトップページの28%とモバイルページの20%に存在し、この割合は下位ランクのサイトに向かって着実に低下し、デスクトップで11%、モバイルで10%に達します。すべてのランクで、デスクトップページはモバイルよりも高いアンロードハンドラーの使用を示しており、アンロードハンドラーがウェブの長尾全体よりも大規模で複雑なサイトでより一般的であることを示唆しています。これは上位サイトがアナリティクス、広告、およびアンロードハンドラーを登録するレガシーライフサイクルパターンに大きく依存しているためと考えられます。

ウェブサイトがbfcacheの対象外カテゴリに入る別の一般的な理由は、`Cache-Control: no-store` ディレクティブの使用です。このキャッシュ制御ヘッダーは、ブラウザ（および中間キャッシュ）にリソースのコピーを保存しないよう指示し、コンテンツがリクエストのたびにサーバーからフェッチされることを保証します。

図7.22. `Cache-Control: no-store` を使用しているサイトの割合。

現在23%のサイトが `Cache-Control: no-store` を使用しており、2024年[306]の21%から増加しました。この増加は、認証済みおよびパーソナライズされた体験の増加、より厳格なセキュリティまたはコンプライアンス要件、特にbfcacheの適格性に関して `Cache-Control: no-store` のパフォーマンスへの影響を低減したブラウザの動作の進化を反映している可能性があります。

歴史的にすべてのブラウザが `Cache-Control: no-store` をbfcacheを避ける理由として扱ってきましたが、Chromeは安全な場合に一部の `no-store` ページのbfcacheを許可する場合があることに注意してください。FirefoxとSafariを含む他のブラウザは一般的に `Cache-Control: no-store` をbfcacheブロッカーとして扱い続けています。

固定画像サイズ

画像はCumulative Layout Shift（CLS）の最も一般的な原因の1つで、ブラウザが事前にどれだけのスペースを確保すべきか分からない場合に発生します。画像が明示的な寸法なしに読み込まれると、ブラウザは最初に画像の高さがゼロであるかのようにページをレイアウトし、画像の読み込みが完了した後に周囲のコンテンツをシフトします。

これを防ぐには、画像は常に `width` と `height` 属性を介して、またはCSSの `aspect-ratio` を使用して本質的な寸法を定義する必要があり、ブラウザは画像がフェッチされる前に正しい量のスペースを割り当てることができます。

306. *https://almanac.httparchive.org/ja/2024/performance#backforward-cache-bfcache*

62%

図7.23. 少なくとも1つの画像に明示的な寸法を設定していないモバイルページの割合。

2025年、明示的な寸法がない画像によるレイアウト不安定性のリスクを抱えたページが依然として大きな割合を占めています。モバイルでは62%のページが少なくとも1つの画像に寸法を設定しておらず、2024年の66%から改善されており、CLS対応の画像慣行の漸進的な採用を示しています。

デスクトップページは同様だがわずかに悪いパターンを示しており、2025年に65%が影響を受けており、2024年の69%から低下しています。下降傾向は前向きですが、大多数のページはまだブラウザがレイアウト時に画像サイズを推測する必要があり、画像をCLSへの最も持続的で防止可能な貢献者の1つにしています。

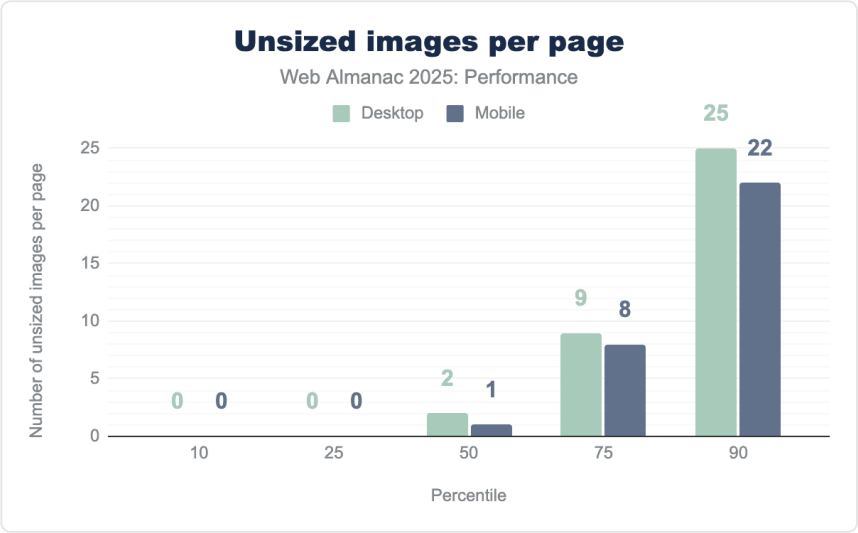


図7.24. ページあたりの寸法未指定画像。

ウェブページあたりの寸法未指定画像の中央値は2枚です。90パーセンタイルでは、この数字はデスクトップで26枚、モバイルで23枚に急増します。寸法未指定の画像はレイアウトシフトのリスクを高めます。ただし、CLSへの実際の影響は、画像のサイズと、特にシフトがビューポートに影響する場合に読み込まれたときにコンテンツがどれだけシフトするかの両方に依存します。CLSは影響フラクション（ビューポートのどれだけが影響を受けるか）と距離フラクション（要素がどれだけ動くか）に基づいて計算されるため、大きな画像やページの上部に近いシフトほどCLSへの貢献が大きくなる傾向があります。

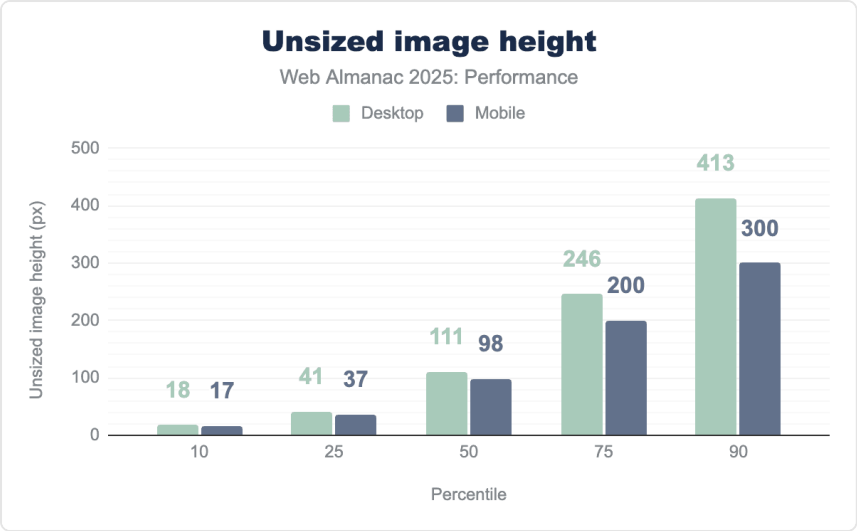


図7.25. 寸法未指定画像の高さ。

寸法未指定の画像は高いパーセンタイルではるかに高くなります。中央値では、寸法未指定画像はデスクトップとモバイルの両方で約100pxの高さですが、90パーセンタイルではデスクトップで約413px、モバイルで300pxに成長します。高い寸法未指定画像がビューポートに表示される場合、特にCLSを増加させます。これは読み込まれたときにより大きな縦方向のレイアウトシフトを引き起こすためです。ウェブページは縦方向にスクロールするため、画像の高さがないことは幅がないことよりもCLSへの影響がはるかに大きいです。

フォント

ブラウザは、カスタムウェブフォントがまだダウンロード中の間、フォールバック（システム）フォントを使用してテキストを最初に表示することがよくあります。この一時的なレンダリングは、Cumulative Layout Shift（CLS）スコアに悪影響を与える可能性があります。カスタムフォントが最終的に読み込まれると、ブラウザはフォールバックフォントを置き換えます。これによりテキストのサイズと間隔が変わり、コンテンツのシフトが起こります。

図7.26. ウェブフォントを使用しているモバイルページの割合。

モバイルページの大多数87%が少なくとも1つのウェブフォントを使用しています。このカスタムタイポグラフィの広範な使用は、ダウンロードと適用を必要とし、レイアウトシフト

の可能性を大幅に高めます[307]。

フォントによるレイアウトシフトを効果的に最小化するには、理想的にはリソースヒントを使用して、できるだけ早く重要なフォントを読み込むことが重要です。フォントが最初のレンダリングの前または非常に近くに読み込まれると、ブラウザはすぐに正しいフォントでテキストを表示できます。これにより、レイアウトシフトの一般的な原因であるデフォルトフォントからのスワップが防止されます。現在のデータは、この機会がしばしば見逃されていることを示しています。

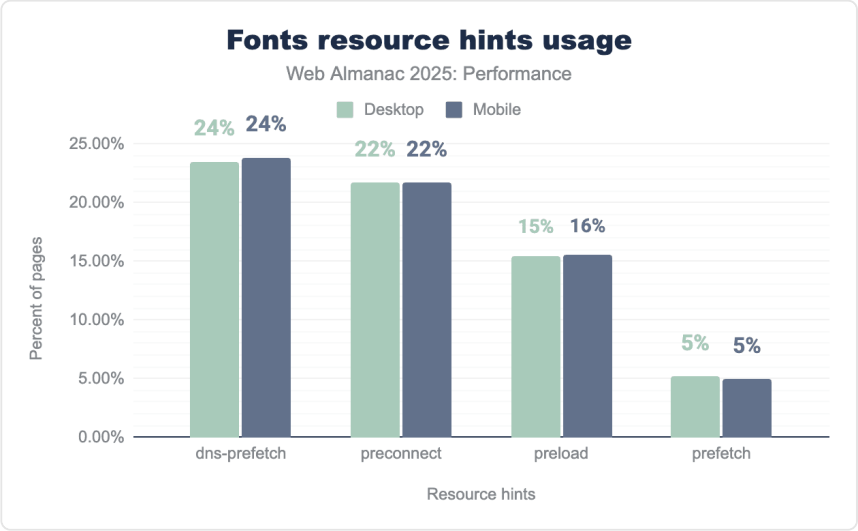


図7.27. フォントリソースヒントの使用状況。

フォントリソースヒントの使用状況はデスクトップとモバイルでほぼ同様です。約24%のページがdns-prefetchを使用し、22%がpreconnectを使用しており、これは主に接続セットアップ時間の削減に役立ちます。Preloadはレンダリング中にフォントを早期に利用可能にする良い方法ですが、ページの15〜16%でしか使用されていません。さらに少ないページ（約5%）がprefetchを使用していますが、これは一般的に初期ページ読み込み中に必要なフォントにはあまり役立ちません。これは、リソースヒントのより的を絞った使用によってフォントのパフォーマンスを改善する大きな機会があることを示唆しています。

非コンポジットアニメーション

非コンポジットアニメーション[308]は、レイアウトに影響するプロパティを変更し、レンダリングが始まった後に周囲のコンテンツを動かすリフローを引き起こすため、レイアウトシフ

307. https://web.dev/articles/optimize-cls#web-fonts
308. https://developer.chrome.com/docs/lighthouse/performance/non-composited-animations

トに貢献します。`transform` や `opacity` などのコンポジットプロパティを使用したアニメーションは、レイアウトを変更せずに視覚的な外観を更新することでこれを回避し、視覚的安定性においてより安全です。

40%

図7.28. 非コンポジットアニメーションを持つモバイルページの割合。

非コンポジットアニメーションは依然として一般的で、モバイルページの40%とデスクトップページの44%に登場しています。

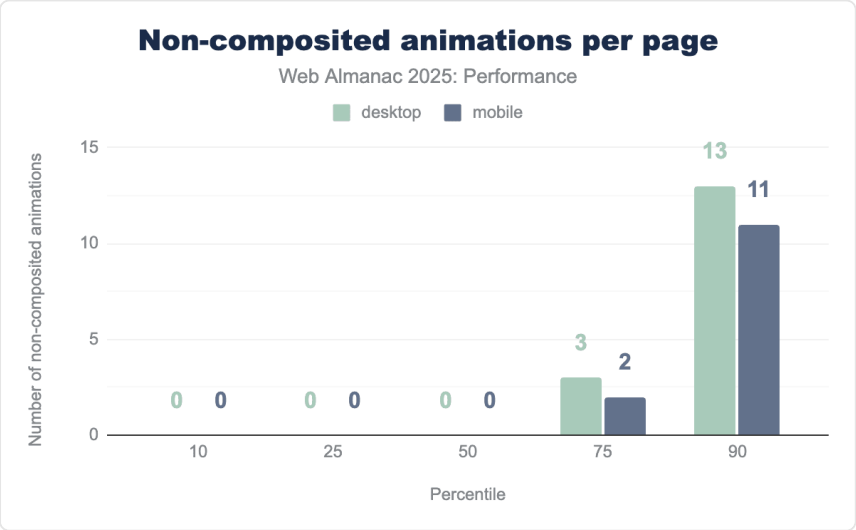


図7.29. ページあたりの非コンポジットアニメーション。

非コンポジットアニメーションの影響は主に高いパーセンタイルで現れ、75パーセンタイルで使用率が増加し、90パーセンタイルではデスクトップで13アニメーション、モバイルで11アニメーションに急増します。

視覚的安定性の結論

ウェブ全体の視覚的安定性は特にモバイルデバイスで年々大幅に進歩しています。ほとんどのページは今や最小限の予期しない動きで安定した体験を提供しており、ベストプラクティスの採用が改善されていることを反映しています。ただし、特にデスクトップで約20〜30%のページがまだ「良好」なCLSを達成していないことから、継続的な改善と最適化の余地が

あります。

漸進的な改善にもかかわらず、寸法未指定の画像は依然として一般的で、フォントの読み込みパターンはまだレイアウトシフトの機会を生み出しており、多くのサイトが既知のCLS軽減策を完全に実装していないことを示唆しています。明示的な画像サイズ設定、重要なフォントのプリロード、コンポジットアニメーションの使用などのシンプルなベストプラクティス[309]を採用することで、ページは視覚的安定性の改善に役立てることができます。

Early Hints

Early Hints[310]は、リクエストされたページを読み込むために必要なリソースについてブラウザに早期シグナルを提供します。

Early Hintsは、リクエストされたページがまだ準備中の間にサーバーからブラウザに送信されます。これにより、ブラウザはリクエストされたページが返される前に、楽観的に他のドメインへのプリコネクトやアセットのプリロードを開始できます。

これにより、Early Hintsは現在リクエストされているページの読み込みパフォーマンスに絶対的な影響を与えることができます。HTMLがブラウザに返るのを待ち、パーサーがメインCSSファイルやLCPアセットのリンク（またはプリロードリンク）を見つけるのではなく、HTMLがブラウザに返される前にそれらのアセットのフェッチを開始できると考えてみてください。これにより、単一の描画でほぼ完璧にレンダリングされたFCPが可能になります。

Early Hintsは `crossorigin` 属性とCSPヘッダー情報も含むことができるため、セキュリティ上の理由から[311]HTTP/2以上でのみ使用することが推奨されます。

309. *https://web.dev/articles/optimize-cls*
310. *https://developer.mozilla.org/docs/Web/HTTP/Reference/Status/103*
311. *https://www.rfc-editor.org/rfc/rfc8297#section-3*

Early Hintsの使用状況

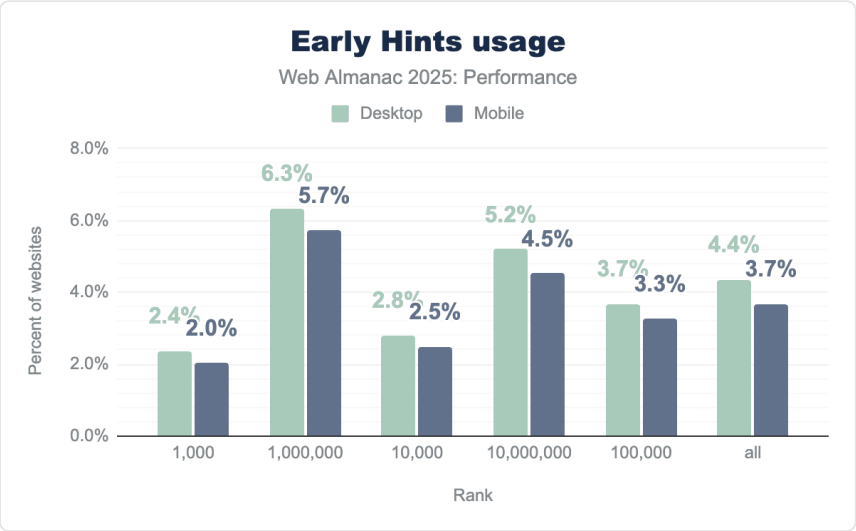


図7.30. Early Hintsの使用状況。

採用はまだ本格化していないことがわかります。すべてのグループで使用率は非常に低く、上位1,000,000サイトのデスクトップでやっと6%を超える程度で、他のほとんどのグループは5%をはるかに下回っています。

これはおそらくEarly Hintsのセットアップと設定の複雑さに関連しています。特定のページのアセットは、ページが完成して送信準備ができる前にサーバーに関連付けられている必要があります。ほとんどのCMSにとってこれは課題となるでしょう。

モバイル/デスクトップの同等性も非常に顕著です。差は1%を超えることはなく、通常は0.5%に近いです。つまり、Early Hintsが実装されている場合、すべてのデバイスタイプで同様に実装されている可能性が高いです。

2025年も使用率は低いままですが、過去3年間で顕著な増加が見られます。

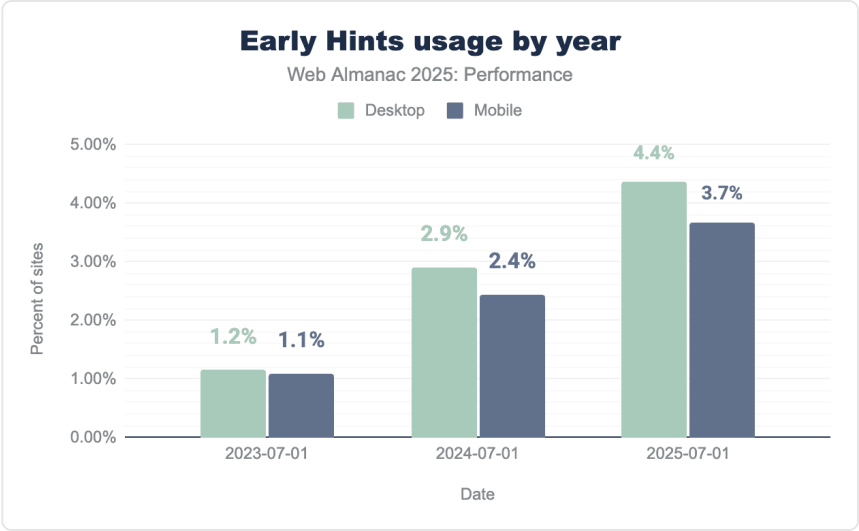


図7.31. 年別Early Hints使用状況。

Early Hintsのサポート

ほとんどのウェブパフォーマンス機能とは異なり、Early Hintsはブラウザだけでなくサーバーのサポートにも依存しています。この公開時点では、`preconnect`はすべてのブラウザでサポートされており、`preload`はSafariを除くすべてのブラウザでサポートされています。

サーバーに関しては、Early HintsはNode、H2O、NGINXで完全にサポートされており、Apacheでは`mod_http2`を使用している場合にサポートされます。

Early HintsはCDNを通じて利用可能で、Fastlyは2020年から[312]、Cloudflareは2021年から[313]、Akamaiは2023年から[314]対応しています。

Speculation Rules

Speculation Rules[315]は、ユーザーが現在のページを表示した後にいずれかのページに移動することを期待して、完全なページを楽観的にプリフェッチまたはプリレンダリングする実験的なブラウザAPIです。これらのアクションはユーザーが現在表示しているページのバック

312. *https://www.fastly.com/blog/beyond-server-push-experimenting-with-the-103-early-hints-status-code*
313. *https://blog.cloudflare.com/early-hints/*
314. *https://www.akamai.com/blog/performance/akamai-103-early-hints-prototype-the-results-are-in*
315. *https://developer.mozilla.org/docs/Web/API/Speculation_Rules_API*

グラウンドで実行されます。現在は主にChromiumベースのブラウザに実装されていますが、Safariはフラグによるプリフェッチの Partial実装があります。

Speculation Rulesは現在のページのパフォーマンスには役立ちませんが、後続ページの読み込みパフォーマンスを大幅に改善し、しばしばほぼ瞬時のページ読み込みに近づけることができます。

このAPIは、より高度な設定オプションでページ向けの `<link rel="prefetch">` と `<link rel="prerender">` を置き換えることを意図しています。なお、Speculation Rules APIはフルページ専用です。個別のアセットに対しては、引き続き `<link rel="prefetch">` を使用する必要があります。

Speculation Rulesの使用状況

以下のチャートはSpeculation Rulesを含むホームページの割合を示しており、興味深いことがわかります。上位1,000サイトでのSpeculation Rulesの使用率は非常に低く、デスクトップで3%、モバイルで5%に過ぎません。各グループが増えるにつれて使用率は上昇しますが、上位1,000,000サイトでもモバイルとデスクトップ合わせて15%にとどまります。最後のグループ、上位10,000,000サイトになってようやく、デスクトップ24%、モバイル25%へと急激に上昇します：

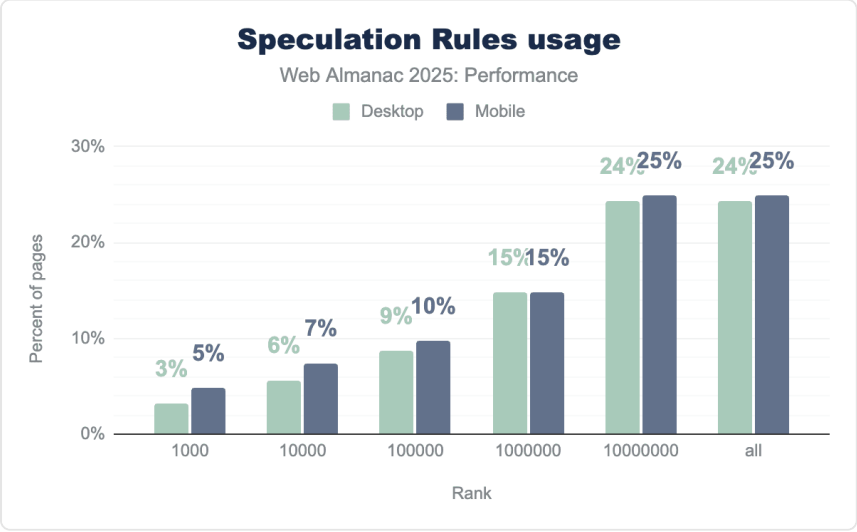


図7.32. Speculation Rulesの使用状況。

これはSpeculation Rulesの設定の複雑さに関連している可能性があります。ユーザーの正確な意図は決してわからないため、サイトはページをプリフェッチまたはプリレンダリングす

る際に慎重である必要があります。フェッチされたが使用されないものはすべて無駄になります。そのため、eコマースサイトのような大規模サイト、特に多数のカテゴリやメニューオプションが直接ジャンプできる大規模サイトでは、Speculation Rulesを適切に設定するのが難しい場合があります。レガシーまたは独自CMSへの実装も複雑になる可能性があります。

逆に、Speculation Rulesはインターネットの大きなシェアを持つ WordPress[316] に組み込まれるようになり、これがロングテールでの高い採用率を説明するかもしれません。

また、モバイルとデスクトップの使用率の同等性も注目に値します。差は1%を超えることはほとんどありません。つまり、Speculation Rulesが実装されている場合、すべてのデバイスタイプで同様に実装されている可能性が高いです。

結論

今年のデータの分析は、よりレスポンシブになりつつあるが、最適化がまだ複雑なウェブの姿を描き出しています。ウェブの使用感における明確な進歩が見られます。モバイルのインタラクティビティは大幅に改善され、スマートフォンとデスクトップコンピュータ間のパフォーマンスギャップがついに縮まり始めています。これは、Interaction to Next Paint（INP）のような新しい指標への業界の注力が功を奏しており、開発者がユーザーにとって最も重要なインタラクションを優先しようとしていることを示しています。

しかし、ウェブの異なるセグメントが新しい標準を採用する方法における「パフォーマンスの分断」も観察されます。例えば、最も人気のあるサイトが複雑なJavaScriptの手動最適化によってインタラクティビティ（INP）の改善をリードしていることがわかりました。これに対して、Speculation Rulesのような新しい標準は、トップではなく、WordPressのような人気CMSのプラットフォームレベルの統合によって推進されるウェブの「ロングテール」で最高の採用率を示しています。これは、パフォーマンスの将来が個々の手動作業への依存を減らし、ウェブを構築するツールに組み込まれたスマートなデフォルトへの依存を増やす可能性があることを示唆しています。

これらの進歩にもかかわらず、ウェブパフォーマンスの基本は依然として課題を突きつけています。高度な指標が改善される一方で、根本的な問題は依然として残っています。モバイルページの約40%が依然として視覚的不安定性のリスクのある非コンポジットアニメーションを使用しており、大多数のページが画像の正しいサイズ指定や重要なフォントをスムーズに読み込むために必要なリソースヒントを欠いています。これは、フレームワークが複雑なJavaScriptの管理を助けてくれる一方で、良好なウェブパフォーマンスを確保するためのより単純なベストプラクティスを見落としがちであることを示唆しています。

316. *https://make.wordpress.org/core/2025/03/06/speculative-loading-in-6-8/*

最後に、測定そのものの状況が成熟しています。より多くのブラウザがINPのような現代的な指標のサポートを拡大するにつれ、クロスブラウザ比較がより一貫したものになります。今後を見据えると、開発者の目標はトップレベルのスコアを超えて、可能性と実践のギャップを埋め、トップサイトが使用する手動最適化と現代のウェブの自動化ツールの両方を活用して、すべてのユーザーに信頼できる体験を提供することです。

著者

Himanshu Jariyal

himanshujariyal himanshujariyal https://medium.com/@him_jar

HimanshuはMicrosoftのBingパフォーマンスチームのシニアソフトウェアエンジニアです。実ユーザーのパフォーマンス計測・分析と、大規模な本番システムの最適化を専門としています。

Prathamesh Rasam

25prathamesh prathamesh-rasam

Prathamesh Rasamは10年以上にわたって大規模なウェブ・モバイルシステムに携わってきたウェブパフォーマンスアーキテクトです。ウェブパフォーマンスに関する公演者として活動し、リアルタイムのウェブ・アプリパフォーマンスモニタリングプラットフォームを大規模に構築しています。

Humaira

hfhashmi

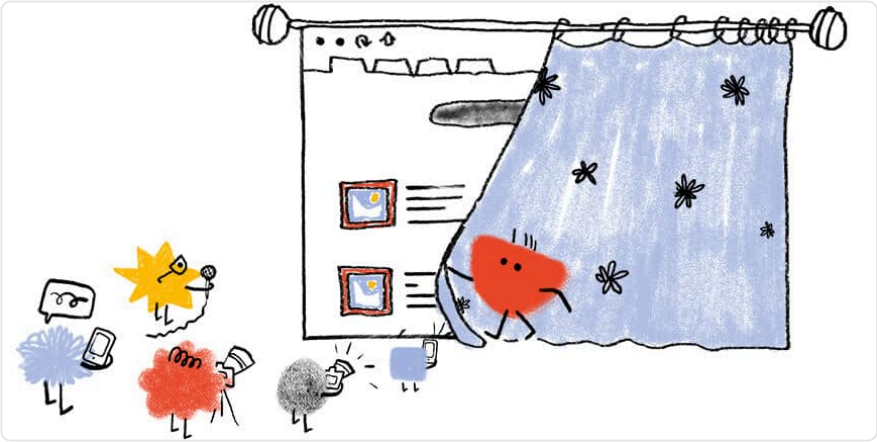
HumaiiraはUCデービス校のコンピュータサイエンスの博士課程学生です。ネットワーク計測、ポリシー、プライバシーの交差点に関する研究を行っています。

Aaron T. Grogg

@aarontgrogg.com aarontgrogg aarontgrogg https://aarontgrogg.com/

Aaron T. Groggは1998年にウェブ開発のキャリアをスタートしました。その後、標準、セマンティックウェブ、アクセシビリティ、そしてパフォーマンスとユーザーに焦点を当てたデジタル体験の重要性を強く提唱するようになりました。Aaronは新しいウェブ技術とベストプラクティスを発見・共有し、すべての人のためにウェブを前進させることに専念しています。

部 II 章 8
プライバシー



Rumaisa Habib、 Vinod Tiwari と Nurullah Demir によって書かれた。
Jannis Rautenstrauch によってレビュ。
Max Ostapenko による分析 と 編集。
Sakae Kotaro によって翻訳された。

はじめに

ウェブはデジタルサービスの主要なインターフェースであり、数十億のユーザーが日々これらのシステムと交流することで、膨大なデータの発生源となっています。その結果、訪問者に関するデータを収集する行為であるウェブサイトトラッキングは、現代のウェブエコシステムの基本的な構成要素となっています。このデータ収集の目的は多岐にわたり、アプリケーションのパフォーマンスや機能の改善から、ターゲット広告やマーケティング分析の実現まで様々です。

しかし、このデータ収集の規模は深刻なプライバシー上の懸念を引き起こしており、技術的[317]・政策的[318]な議論の場でも広く取り上げられ、研究[319]の主要なテーマにもなっています。開発者がHTTP Cookieやブラウザフィンガープリンティングなどの様々な技術を用いてユーザーを追跡する一方で、ブラウザベースの制限、法規制対応ツール、プライバシー強化拡張

317. *https://www.w3.org/TR/tracking-compliance/*
318. *https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng*
319. *https://pulse-of-cybersecurity.com/topics?sortBy=total-papers&sortOrder=desc&page=1&pageSize=21&search=web&topic=Web+Tracking+and+Browser+Fingerprinting&conferences=%5B%5D*

機能といったプライバシー保護措置も拡大しています。

本チャプターでは、ウェブプライバシーの現状について技術的な概要を提供します。一般的なトラッキング手法の採用状況を分析し、トラッキングを防ぐための措置の普及状況を調査することで、現在のユーザーデータ収集の全体像をデータに基づいて考察します。

オンライントラッキング

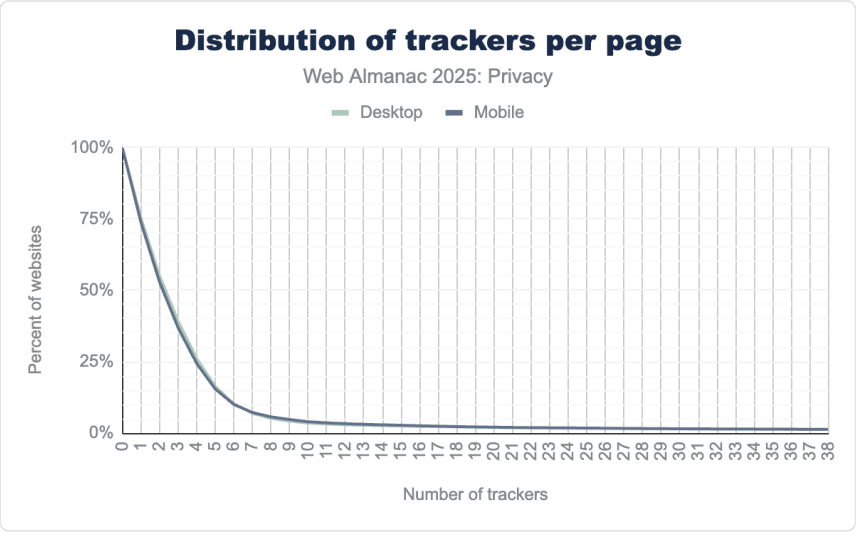


図8.1. ページあたりのトラッカー分布

本分析では、WhoTracks.Me[320]の人気サードパーティトラッカーカタログを使用して、ウェブページに存在するトラッカーを特定しています。分析を保守的に行うため、WhoTracksMeのカテゴリのうち `advertising`、`pornvertising`、`site_analytics`、`social_media` のみをトラッカーとしてカウントしています。この手法により、各ウェブページについてドメインレベルで個別のサードパーティトラッカーを特定できます。なお、報告される数値はHTTPリクエストの総数ではなく、ユニークなドメイン数を表している点に注意が必要です。

320. *https://www.ghostery.com/whotracksme/*

75%

図8.2. 少なくとも1つのトラッカーを持つウェブサイト。

全ウェブページの75%（デスクトップ75%、モバイル74%）に少なくとも1つのサードパーティトラッカーが存在することが確認されました。デスクトップページの55%は2つのトラッカーを含み、39%は3つのトラッカーを含んでいます。6つ以下のトラッカーの設定はデスクトップページでより多く見られる一方、7つ以上のトラッカーはモバイルページでより多く確認されています。

ステートフルトラッキング

トラッキングの仕組みはステートフルとステートレスに分類されます。Cookieやローカルストレージなどのステートフルなアプローチは、識別情報をユーザーのデバイスに保存します。一方、フィンガープリンティングのようなステートレスなアプローチは、実行時に固有の特性からこの情報を推測します。

サードパーティトラッキングサービス

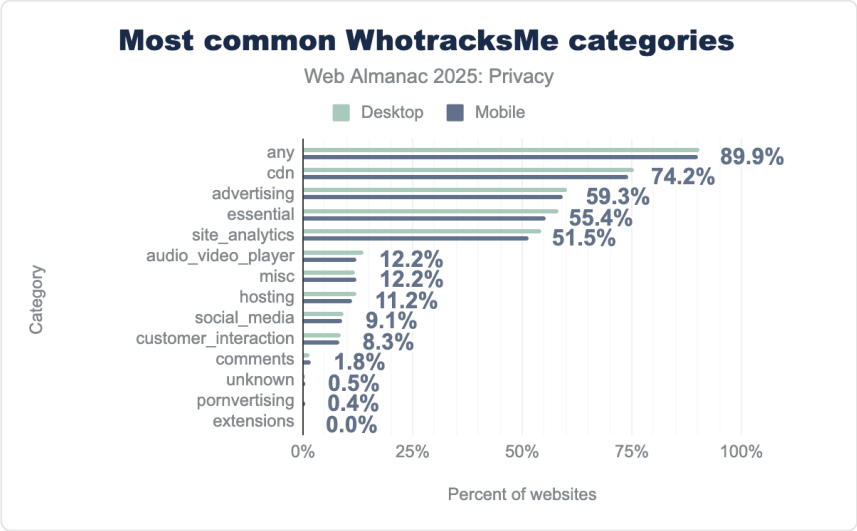


図8.3. 最も一般的なWhoTracksMemカテゴリ

ここでは、WhoTracksMeのすべてのカテゴリを考察します。大半のウェブページが、コン

テンツデリバリネットワーク（CDN）および広告に分類されるドメインに接続していることが確認されました。CDN関連ドメインはウェブページの74%に存在し、続いて広告関連ドメインが59%に見られます。また、ウェブページの55%には必須ドメイン（Google Tag Managerなど）が含まれ、52%には解析ドメイン（Google Analyticsなど）が含まれています。少数の主要プレイヤーへのこの高い集中は、ウェブプライバシーのベースラインを効果的に設定しており、大多数のユーザーデータが少数の支配的なプラットフォームを通じて流れる構造になっています。

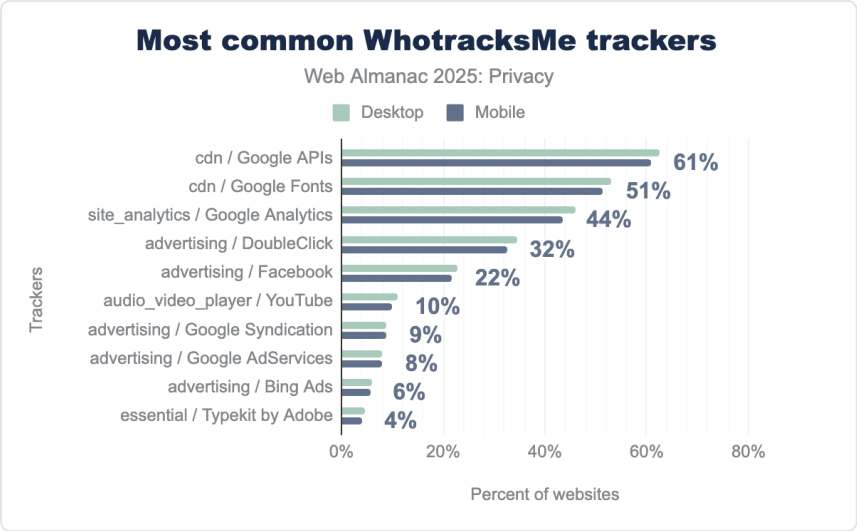


図8.4. 最も一般的なWhoTracksMeトラッカー

個別トラッカーの分析では、GoogleとFacebookがトラッキングサービスの大半を占めていることが明らかになりました。これらはウェブ上で最も存在感のあるトラッカーであり、Googleはウェブページの61%に、Facebookは少なくとも22%のウェブページに存在しています。次いでBing（6%）、Adobe（4%）が続きます。

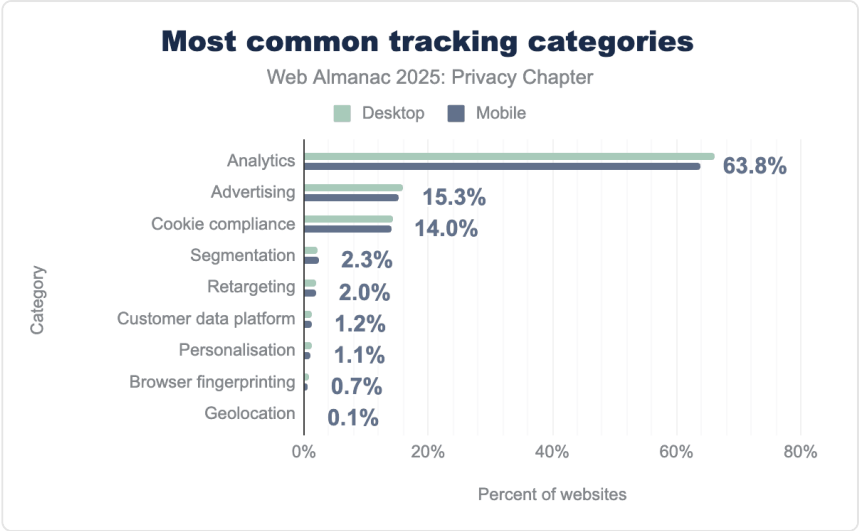


図8.5. 最も一般的なトラッキングカテゴリ

さらに、これらのサービスを機能別に分類すると、解析がデスクトップウェブページの64%に登場して全体をリードしています。広告（15%）とCookie同意ツール（14%）がこれに続き、パフォーマンス監視がデータ収集の主な目的であり続けていることを示しています。セグメンテーションやリターゲティングなどのより専門的なトラッキング手法は、それぞれサイトの3%未満と格段に少ない状況です。

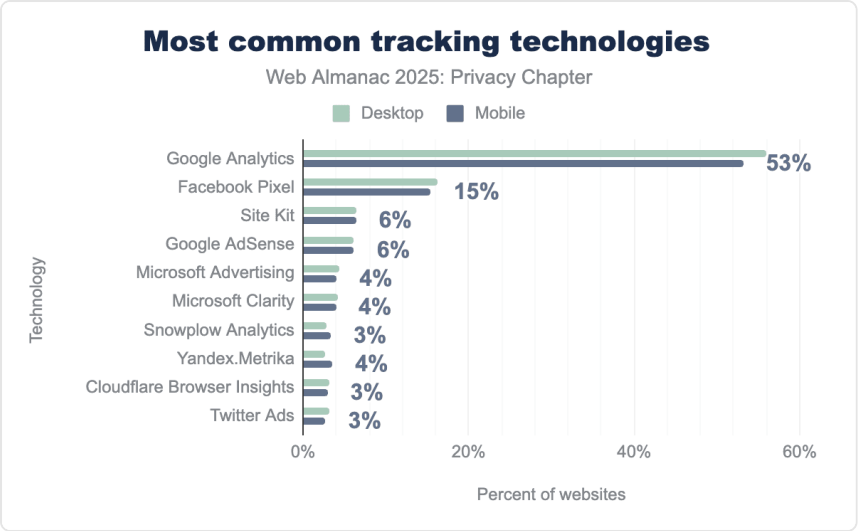


図8.6. 最も一般的なトラッキング技術

トラッキングは、ウェブページ上のユーザー行動の把握から複雑な広告プロファイルの構築まで、様々な文脈で行われます。ウェブユーザーのトラッキングに最も多く使われている技術は、Google Analytics（53%）とFacebook Pixel（16%）であることが分かりました。これらの市場リーダーを超えると採用率は急激に低下し、GoogleのSite Kit（6.41%）とAdSense（6.18%）が次のティアを形成しています。Microsoftなどの他のプレイヤーも、AdvertisingとClarityがそれぞれウェブサイトの約4%に存在するなど、一定ながらも小さいシェアを維持しています。

サードパーティCookie

サードパーティCookieの利用は、ウェブユーザーのトラッキングとターゲティングに効率的な手法です。サードパーティはユーザートラッキングにCookieを活用しており、継続的な批判にもかかわらず、ウェブ上で一般的な手法であり続けています。Googleのような一部のベンダーはサードパーティCookieの廃止[321]を発表し（その後撤回[322]しましたが）、依然としてトラッキングの重要な手法であり、トラッキング目的で使われるサードパーティCookieの大部分を占めています。

321. *https://support.google.com/google-ads/answer/14762010*
322. *https://privacysandbox.google.com/blog/privacy-sandbox-update*

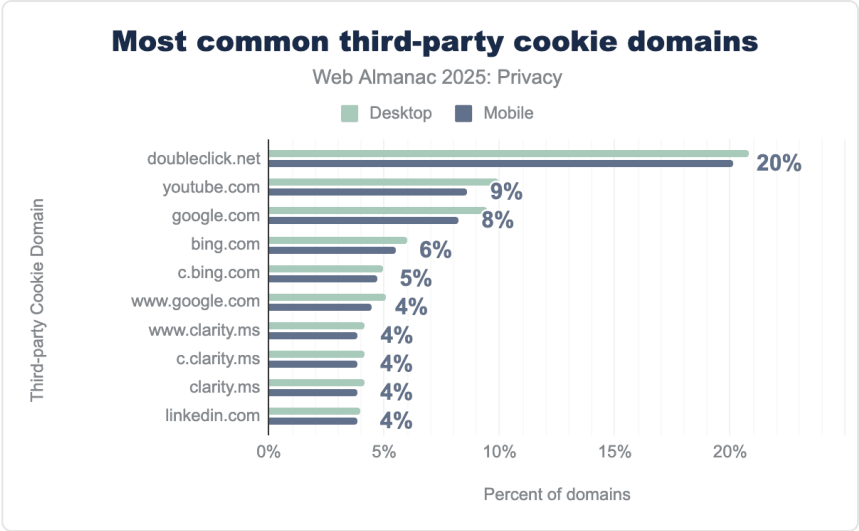


図8.7. 最も一般的なサードパーティCookieドメイン

分析の結果、`doubleclick.net` がデスクトップサイトの20%に登場する最も一般的なサードパーティCookieドメインであり、続いて `youtube.com`（9%）と `google.com`（8%）が続くことが分かりました。全体として、Googleのエンティティが上位を占める中、Microsoftの `bing.com` と `clarity.ms`、そして `linkedin.com` が最も重要な代替サードパーティCookie設定元となっています。

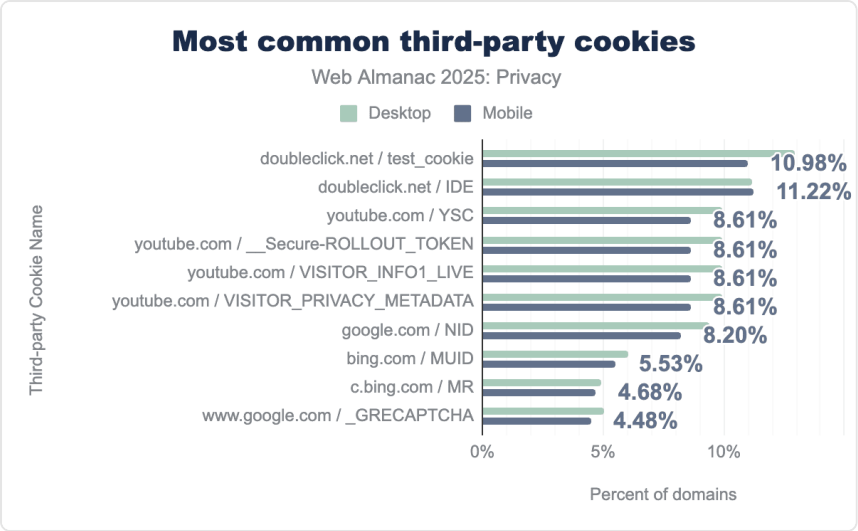


図8.8. 最も一般的なサードパーティCookie

ファーストパーティCookie

次の図は最も一般的なファーストパーティCookieを示しています。これらのCookieはファーストパーティのコンテキストで設定されていますが、その名前からトラッキング目的で主に使用されていることが分かります。`_ga` CookieはGoogle Analyticsによってウェブページの46%に設定され、`_gid` は18%に登場しており、続いて `gcl_au` がウェブページの16%に見られます。これらのCookieの正確な目的は検証していませんが、GoogleはそれらのCookieの意図された機能を公開[323]しています。 もう一つの人気ファーストパーティCookieは `_fbp` で、Metaによってウェブページの14%で使用されています。MetaはMetaピクセルとともにファーストパーティCookieを使用するオプションを広告主に提供[324]しています。サードパーティのコンテキストで観察された結果と同様に、ファーストパーティCookieのコンテキストでもGoogleとMetaがトラッキングの主要エンティティであり続けています。

ウェブ上でのCookieの使用は依然として主にトラッキング目的です。機能的な例外として、`PHPSESSID` はPHPアプリケーション用のユニークなセッションIDをページの12%に保存し、`XSRF-TOKEN` はクロスサイトリクエストフォージェリ対策のセキュリティを担いウェブページの6%に見られます。

323. *https://business.safety.google/adscookies/*
324. *https://www.facebook.com/business/help/471978536642445?id=1205376682832142*

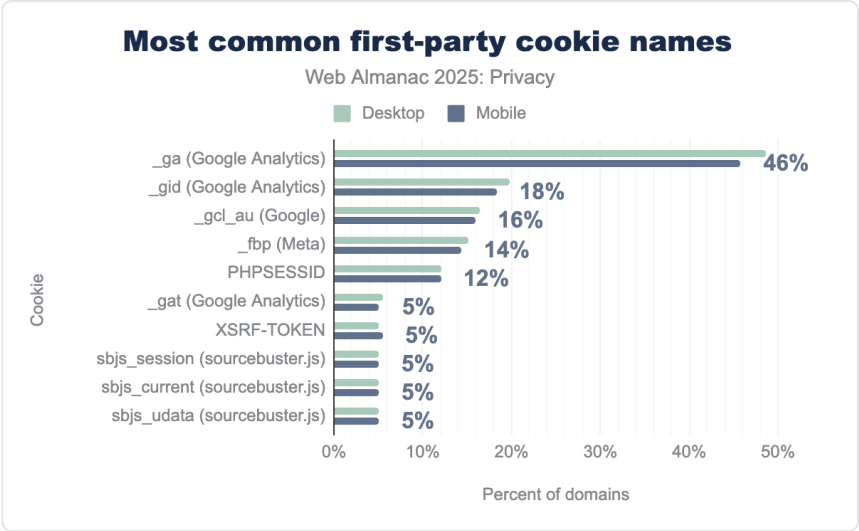


図8.9. 最も一般的なファーストパーティCookie名

Cookieチャプターでは、Cookieの詳細と使用傾向についてさらに詳しく説明しています。

ステートレストラッキング

ステートレストラッキングは、ユーザー識別子をブラウザに状態として保存するのではなく、その場で生成するプロセスです。これらの識別子は通常、対象ユーザーのデバイスやブラウザから能動的または受動的に収集できる情報を使用して作成されます。複数のデバイスを使用するユーザーのセッションを相関させることは難しいものの、一部のシグナルはデバイスやウェブサイトの機能に固有のものであり、簡単に「ブロック」できないという点で効果的です。

ブラウザフィンガープリンティング

ブラウザフィンガープリンティングは、ウェブサイトがユーザーのブラウザ固有の情報に基づいてユーザーを特定する手法です。この情報には、システムフォント、言語設定、ハードウェア構成など、一見無害に見えるデータポイントが含まれます[325]。これらは個別には少ない情報しか明かしませんが、組み合わせることで特定のユーザーの固有のプロファイル[326]を描くことができます。これらは一般的にHTTPヘッダーやJavaScript APIコールを通じて漏洩します。

325. https://dl.acm.org/doi/abs/10.1145/3543507.3583333
326. https://amiunique.org/

先行研究[327]により、ブラウザフィンガープリンティングはオンライントラッキングにおいて非常に広く使われていることが明らかになっています。その魅力は、ブロックが難しく、ユーザーがシークレットブラウザを使用していても効果的であると言われている点にあります。本レポートでは、ブラウザフィンガープリンティングに使用される最も一般的な技術を特定します。

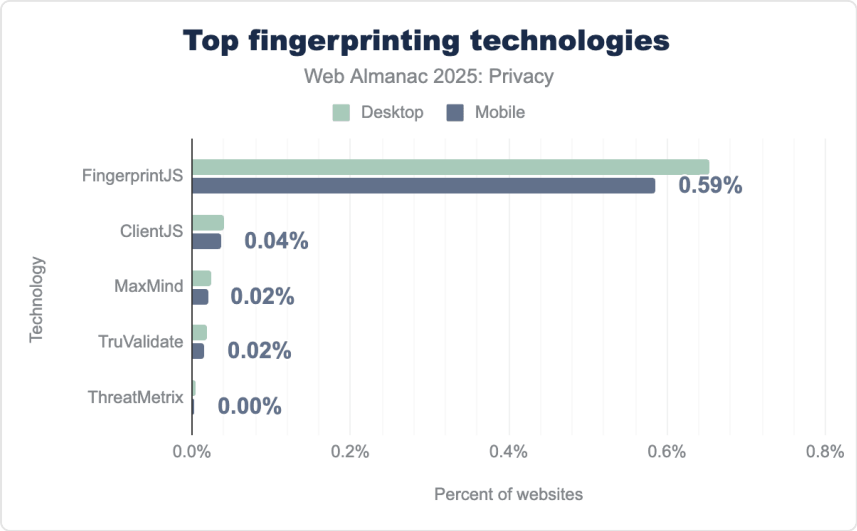


図8.10. 上位のフィンガープリンティング技術

特筆すべきことに、FingerprintJS[328]ライブラリはブラウザフィンガープリンティングを実施する最も人気のあるツールであり続けており、他のライブラリを大きく引き離しています。FingerprintJSはモバイルアクセスのウェブサイトの0.59%で使用されており、0.04%に留まるClientJS[329]（次に人気の高い技術）を大幅に上回っています。

FingerprintJSの人気は、ClientJSよりも活発に見えるその活発なオープンソースコミュニティに起因していると考えられます。

トラッキング保護の回避

ブラウザやプライバシーツールがサードパーティトラッカーのブロックにおいてより効果的になるにつれて、トラッキング業界も適応してきました。CNAMEクローキングやバウンストラッキングなどの技術により、トラッカーはファーストパーティリソースを装ったり、中

327. https://dl.acm.org/doi/abs/10.1145/3696410.3714548
328. https://github.com/fingerprintjs/fingerprintjs
329. https://github.com/jackspirou/clientjs

間リダイレクトを使用して従来のブロック手法を回避したりすることができます。これらのアプローチはブラウザがファーストパーティリクエストに置く信頼を悪用するため、検出とブロックがより難しくなります。このセクションでは、クロールデータのリダイレクトチェーンを通じて観察できるバウンストラッキングに焦点を当てます。

バウンストラッキング

バウンストラッキングは、ユーザーが目的地に到達する前に一時的に中間ドメインを経由してリダイレクトされる手法です。このリダイレクト中（ユーザーには気づかれないことが多い）、中間サイトはCookieを設定または読み取ることができ、ファーストパーティとのやり取りのように見えながらもサイトをまたいでユーザーを効果的に追跡できます。これにより、従来のサードパーティCookieブロックを迂回することができます。

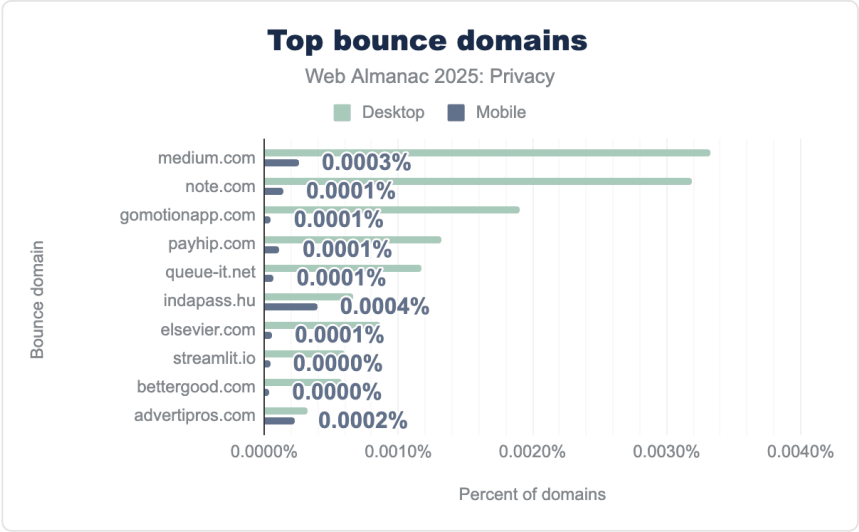


図8.11. 上位のバウンスドメイン

2025年のデータセットでは `medium.com` が0.0003%で引き続き最も一般的なバウンスドメインであり、`note.com` と `indapass.hu` がこれに続きます。前年比で普及率は大幅に低下しており、例えば `medium.com` は0.009%から0.0003%に、`indapass.hu` は0.012%から0.0004%に減少しました。この低下はChromeのバウンストラッキング対策が効果を発揮していることを反映していると考えられます。上位ドメイン（`queue-it.net`、`payhip.com`、`medium.com`）は正当な機能的サービスであるため、データは観察された動作の大半が秘密のトラッキングではなく、必要なリダイレクトによるものであることを示唆しています。

プライバシーを改善するブラウザポリシー

ブラウザは、ウェブサイトがサードパーティと共有する情報量に影響を与えるさまざまな仕組みを導入してきました。これらの機能はプロトコルレベルで動作し、ヘッダーを制御し、データの露出を制限し、サイトが外部リソースと通信する方法を標準化します。実際のプライバシーへの影響は、実装、採用状況、そしてサイトがこれらを使用するかどうかによって異なります。

このセクションでは、3つのメカニズムを検討します。従来のUser-Agent文字列に対してより制御された代替手段を提供するUser-Agent Client Hints、サードパーティに渡されるナビゲーションのコンテキスト情報量をサイトが制限できるReferrer Policy、そして広範な展開前にブラウザが新機能を試験するプライバシー関連のOrigin Trialsです。

User-Agent Client Hints

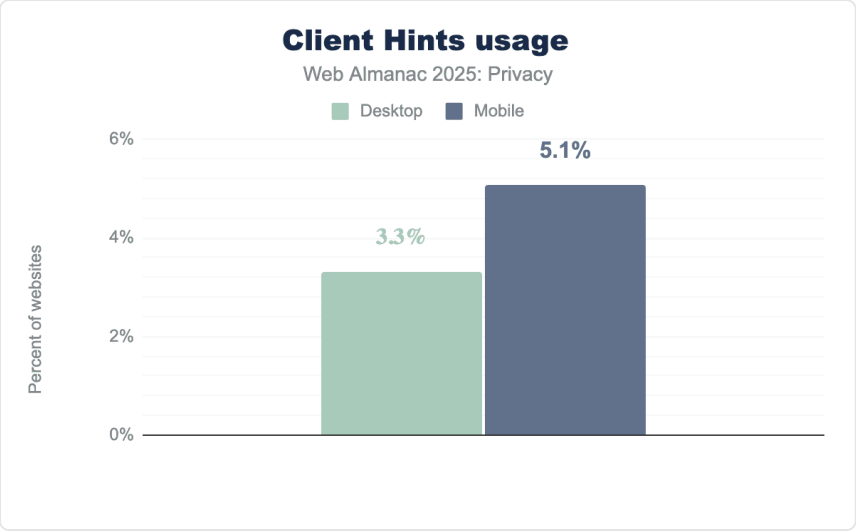


図8.12. Client Hintsの使用状況

User-Agent Client Hintsは、従来のUser-Agent文字列に対してプライバシーを重視した代替手段を提供し、ブラウザがサーバーから明示的に要求された場合のみデバイスとブラウザの情報を共有できるようにします。デフォルトで詳細なフィンガープリントを公開するのではなく、サイトは特定のヒントにオプトインする必要があるため、受動的なデータ漏洩が削減されます。2025年、採用率はデスクトップで3.3%、モバイルで5.1%となっており、モバイルの方が高い理由はレスポンシブデザインシグナルへのニーズが大きいことを反映していると考えられます。この移行は恒久的なものとなっており、Chrome 145が

`UserAgentReduction` ポリシーを廃止したことで、詳細なブラウザやデバイス情報が必要なサイトはUser-Agent Client Hintsへの移行[330]が必要です。

昨年のデータは、サイトの人気度とClient Hintsの使用間に強い相関関係を示していました。上位1,000サイトは15.85%に達する一方、10万位層では約1.6%まで急落しました。今年の方法論ではランク別の内訳がありませんが、全体的な数値から採用は依然として大規模なサイトに集中しており、ロングテールはまだこの標準を採用していないことが示唆されます。

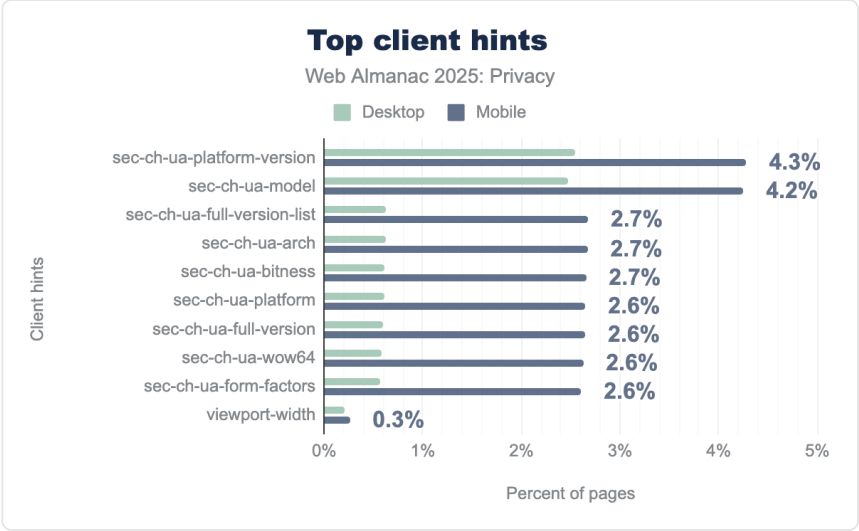


図8.13. 上位のClient Hints

最もリクエストの多いClient Hintsは sec-ch-ua-platform-version で4.28%を占め、互換性判断のためのOS版バージョン検出に使用されます。sec-ch-ua-modelが4.25%で僅差で続きますが、注目すべき偏りがあります。モバイルの使用がデスクトップを大幅に上回っており、デバイスモデルが主にモバイル体験とデバッグに関連していることを考えると理にかなっています。アーキテクチャ、ビット数、フルバージョンリスト、フォームファクターを網羅する残りのヒントは2.60%〜2.67%の間に密集しており、Client Hintsをリクエストするサイトは個々のシグナルを選択するのではなく、複数をまとめてリクエストする傾向があることを示唆しています。

330. https://web.dev/articles/migrate-to-ua-ch

Referrer Policy

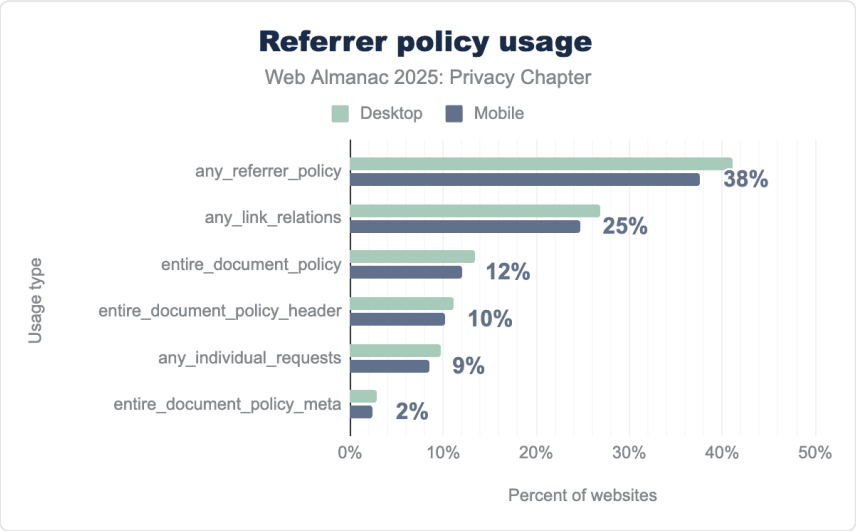


図8.14. Referrer Policyの使用状況

あるウェブサイトから別のウェブサイトのリンクをクリックすると、ブラウザはページのフルURLを含む参照元情報を公開する可能性があります。Referrer Policyは、ウェブサイトのオーナーがこの情報の共有量を制御できる仕組みで、サードパーティがユーザーの閲覧経路について参照できる情報を制限することでプライバシー保護に役立ちます。

Referrer Policyの全体的な採用率は2024年の32%から2025年の37.66%に上昇しており、健全な増加といえます。最も一般的な実装方法は、個別リンクへの `rel="noreferrer"` などのリンクレベルの制御で24.70%を占め、ヘッダーで設定するドキュメント全体のポリシーは10.16%となっています。これは、多くのサイトが一律のルールとしてではなく、選択的にリファラー制限を適用していることを示しています。

メタタグによる実装は2.47%で依然として最も少なく、2024年の2%からほぼ変わらない状況です。これは予想通りの結果で、ヘッダーはページ読み込み前に適用され改ざんが難しいため、セキュリティポリシーには一般的にヘッダーが好まれます。

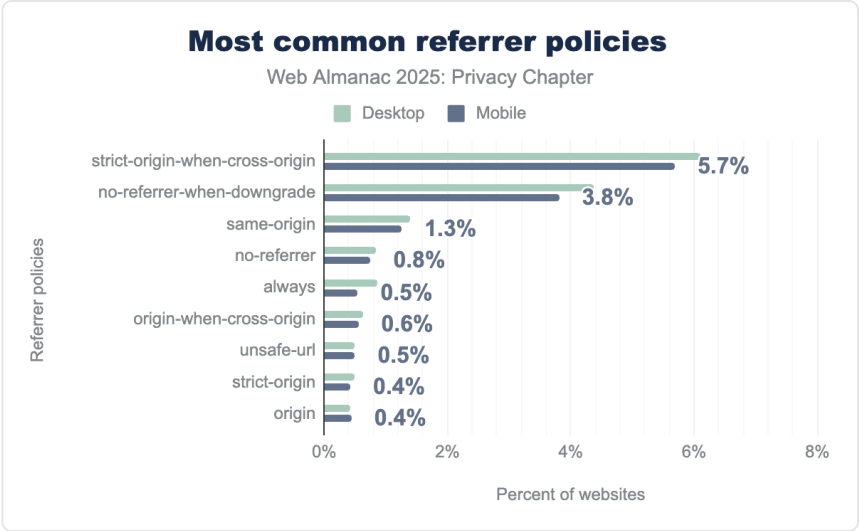


図8.15. 最も一般的なReferrer Policy

今年は最もプライバシーを重視するポリシーで低下が見られました。他のサイトへ移動する際にオリジンは共有するがフルパスを削除する strict-origin-when-cross-origin は7.5%から5.69%に低下しました。同様に、no-referrer-when-downgrade は7.0%から3.81%に落ちました。これらは依然として上位2つのポリシーですが、低下はサイトの一部が設定を緩和したか、実装方法を変更したことを示唆しています。

ポジティブな面としては、same-origin（1.26%）やno-referrer（0.75%）などの真に制限的なオプションは引き続き使用されていますが、採用率は低い状況です。これらのポリシーはサードパーティのサイトと情報を一切共有しないため、プライバシーには理想的ですが、サイトが依存する解析やアフィリエイトトラッキングには制限的な場合があります。

一部のサイトは依然として unsafe-url（0.50%）を指定しており、これはあらゆる接続先にフルURLを公開しますが、この動作はChromeに固有で他のブラウザでは廃止されています。also（0.54%）という無効な値も見られ、ブラウザはこれを無視してデフォルトの strict-origin-when-cross-origin にフォールバックします。これらの値が存在することは、サイトの一部がプライバシーに不親切な設定を意図的に選択しているのではなく、Referrer Policyの設定が誤っているか古くなっていることを示唆しています。

プライバシー関連のOrigin Trials

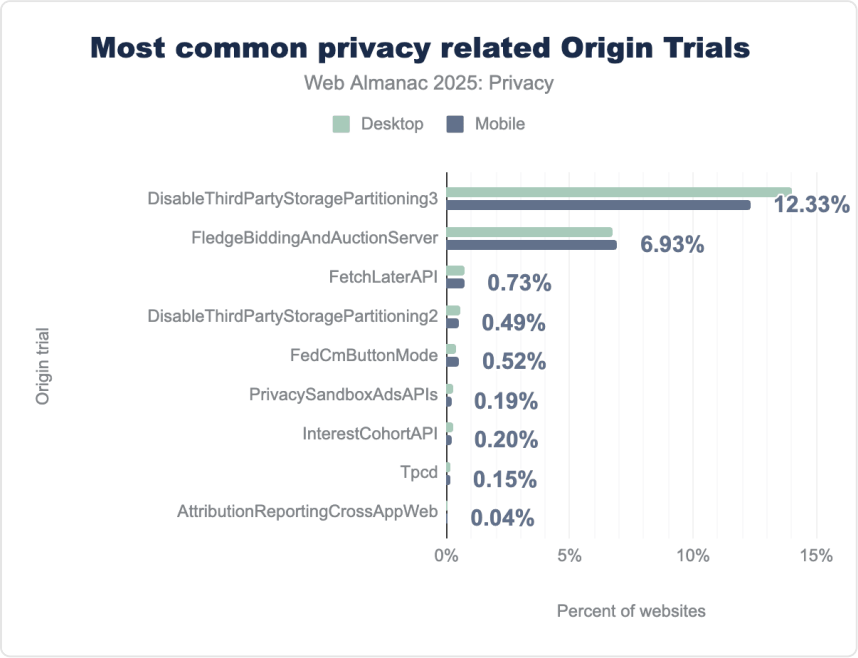


図8.16. 最も一般的なプライバシー関連のOrigin Trials

Origin Trialsにより、ブラウザは完全な展開にコミットする前に実際のウェブサイトで実験的な機能をテストできます。サイトは早期に新機能にアクセスするためにオプトインするか、既存の機能を壊す変更を一時的に遅らせるための廃止トライアルにオプトインすることができます。これらのトライアルはブラウザベンダーが本番環境での機能パフォーマンスに関するデータを収集しながら、開発者が適応する時間を与えるために役立ちます。これから分かるように、プライバシー関連の採用のほとんどは廃止カテゴリに該当しています。

最も広く採用されているトライアルは依然として`DisableThirdPartyStoragePartitioning`で、2024年の10.21%から2025年の12.33%に増加しました（現在3回目のイテレーション）。このトライアルにより、サイトはCookieとストレージをサイトごとに分離するプライバシー機能であるストレージのパーティショニングを一時的にオプトアウトし、開発者がレガシー実装を移行する時間を確保できます。同様に、サイト横断トラッキングなしで興味関心ベースの広告を行うGoogleのPrivacy Sandboxイニシアチブの一部である`FledgeBiddingAndAuctionServer`は、6.62%から6.93%に緩やかに増加しました。

最大の変化は`AttributionReportingCrossAppWeb`で、2.10%から0.04%へと急落しまし

た。これはトライアルが終了したか、サイトがクロスアプリのアトリビューションテストから離れたことを示唆しています。新しいエントリには `FetchLaterAPI`（0.73%）、遅延リクエスト、フェデレーテッドアイデンティティが含まれます。一方、物議を醸したFLoCの前身である `InterestCohortAPI` は0.20%と大きく変わらず、主に残留物として留まっています。

法律と政策

プライバシー規制は、ウェブサイトとユーザーとのやり取りのあり方を形作り続けています。このセクションでは、サイトが同意ダイアログを通じてどのように対応しているか、また Do Not Track や Global Privacy Control などのプライバシーシグナルが有意義な採用を獲得しているかどうかを検討します。

同意ダイアログ

GDPR[331]やCCPA[332]などのプライバシー規制は、個人データを収集・処理する前にユーザーの同意を得ることをウェブサイトに義務付けています。これにより、Consent Management Platform（CMP）によって管理されることが多いCookieの同意ダイアログが、現代のウェブのほぼ普遍的な機能となっています。広告エコシステム全体で同意の収集と伝達を標準化するために、Interactive Advertising Bureau（IAB）はTransparency and Consent Framework（TCF）、US Privacy String（USP）、より新しいGlobal Privacy Platform（GPP）などのフレームワークを開発しました。

これらのフレームワークはユーザーに制御権を与えることを目的としていますが、採用率と実装品質は大きく異なります。一部のサイトはTCFv2に完全に準拠している一方、実装が不完全なサイトや古い標準に依存しているサイトもあります。また、クローラーが米国を拠点としており、TCFの下では非EU訪問者にはCookieバナーの表示が不要なため、実際のTCF使用量はここで計測している値よりも高い可能性があることも注目に値します。

331. *https://gdpr-info.eu/*
332. *https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5*

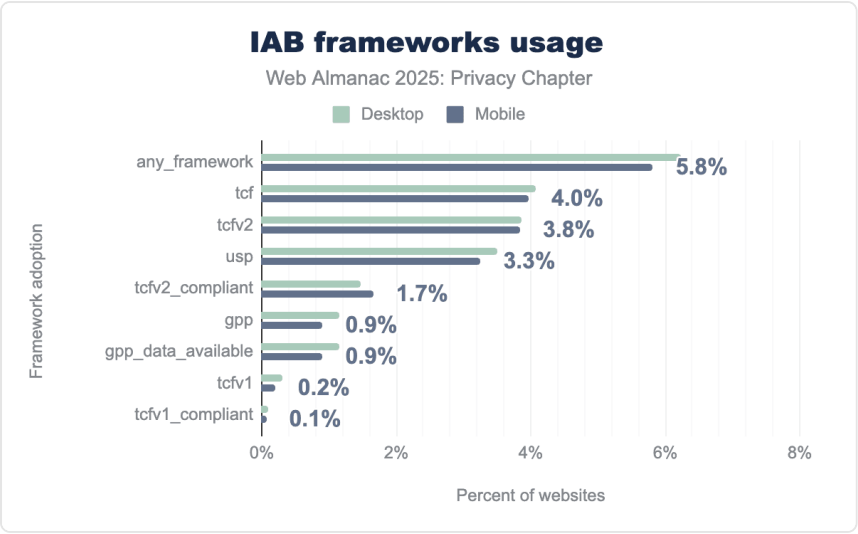


図8.17. IABフレームワークの使用状況

IABフレームワークの全体的な採用率は、モバイルとデスクトップの両方で5.5%をわずかに超える水準で安定しています。TCFは4.0%で引き続き最も広く採用されているフレームワークで、TCFv2がその3.8%を占めています。しかし、完全にTCFv2に準拠しているサイトは1.7%に過ぎず、TCFv2を使用していると主張するサイトの半数未満であり、多くの実装が不完全または不適切に設定されていることを示唆しています。USPは3.3%で安定しており、CCPA対応への継続的な取り組みを反映しています。

廃止されたTCFv1はほぼ消滅しており、0.2%に留まり準拠しているのはわずか0.1%と、業界がv2に移行した可能性を示しています。今年の注目すべき追加はGPPで、IABの新しい統合フレームワークがサイトの0.9%に登場しています。好ましいことに、gpp_data_availableが0.9%と一致しており、GPPを採用したサイトがコードを読み込むだけでなく、実際にユーザーの設定を伝達するために使用していることを意味しています。

前年比で見ると、全体的なフレームワーク採用率は横ばいだった一方、TCFの使用率は4.2%から4.0%にわずかに低下しました。この緩やかな低下はGPPへの早期移行を反映している可能性がありますが、トレンドと呼ぶには早すぎます。コンプライアンスのギャップは続いており、TCFv2のコンプライアンスは1.7%で変わらず、採用だけでは適切な実装を保証しないことを浮き彫りにしています。

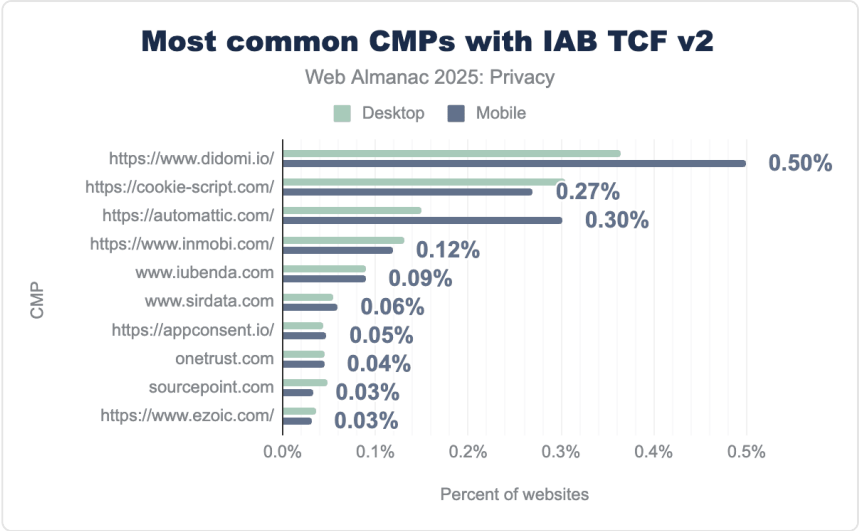


図8.18. IAB TCF v2で最も一般的なCMP

今年はCMPの状況が大きく変化しました。2024年に0.67%でリードしていたAutomatticは2025年に0.30%に低下した一方、Didomiは0.22%から0.50%に上昇してトップの座を獲得しました。Cookie-scriptが0.27%で新規参入し、デスクトップで2位にランクインしました。残りのプロバイダー（InMobi、Iubenda、Sirdata、AppConsent、OneTrust、Sourcepoint、Ezoic）はそれぞれサイトの0.12%未満で、TCFv2のCMP採用が少数の主要プレイヤーに集中していることが分かります。

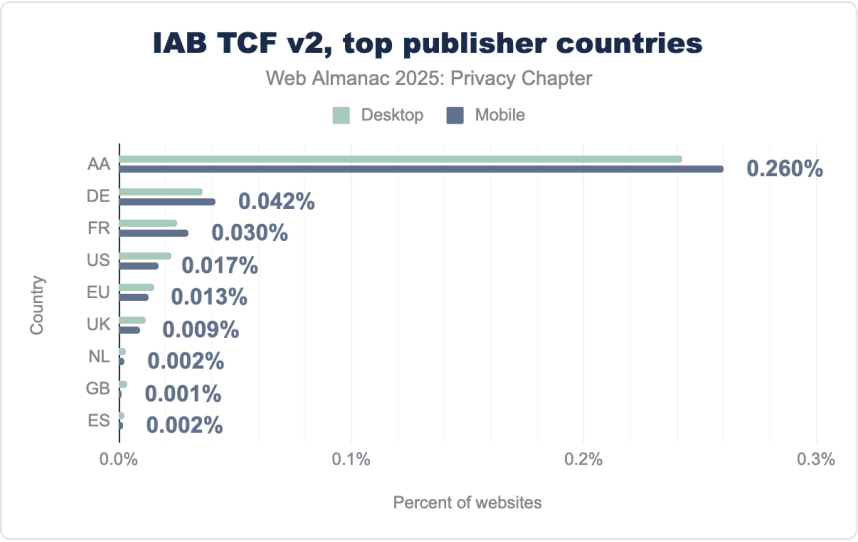


図8.19. IAB TCF v2、上位パブリッシャー国

EUの加盟国の中ではドイツ（0.042%）とフランス（0.030%）がTCFv2パブリッシャーの採用をリードしており、EUの域外ではTCFが義務でないにもかかわらず米国が0.017%で登場しています。最も大きな割合（0.26%）は未定義の国コード「AA」に分類されており、パブリッシャーのメタデータのギャップやCMPの実装の誤りを示しています。GDPRの要件にもかかわらず、欧州のパブリッシャーの間でさえ全体的な採用率は低く、TCFv2が一部の少数のサイトに集中していることを示唆しています。

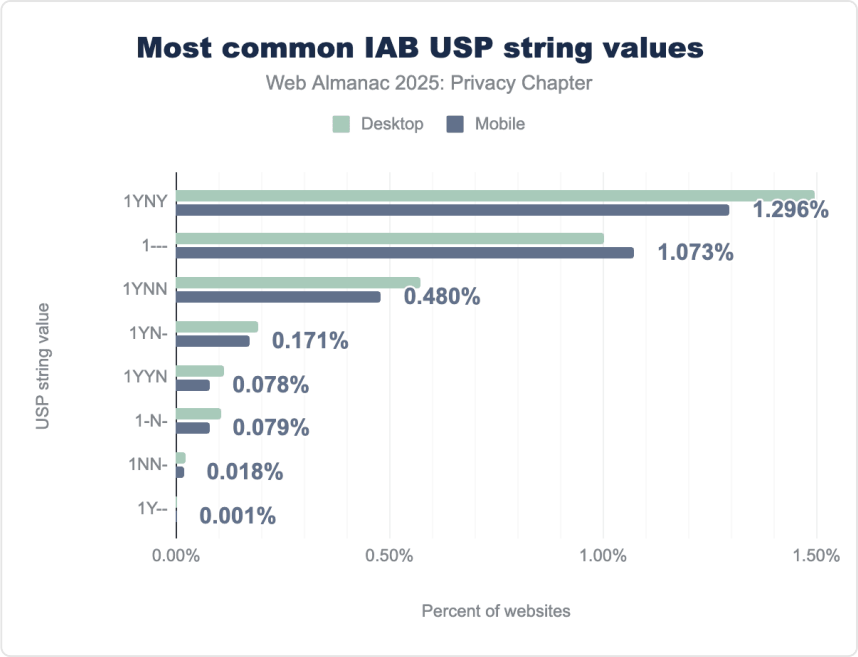


図8.20. 最も一般的なIAB USP文字列の値

最も一般的なUSP文字列は1.296%の1YNYで、通知が行われ、ユーザーがオプトアウトしておらず、サイトがLimited Service Provider Agreementの対象であることを示します。2番目に多い値は1.073%の1---で、意味のあるシグナルを提供しないプレースホルダー文字列であり、多くの実装が不完全またはデフォルト設定であることを示唆しています。`1YYN` を表示するサイトは、新規訪問者をデフォルトでオプトアウト状態にするようCMPを設定しており、これは要件よりも厳格なプライバシーの姿勢です。低い普及率（0.078%）は、ほとんどのサイトが同意を明示的に撤回するまでの間は同意しているとみなすCCPAの標準的なオプトアウトモデルに従っていることを示しています。

Do Not Track

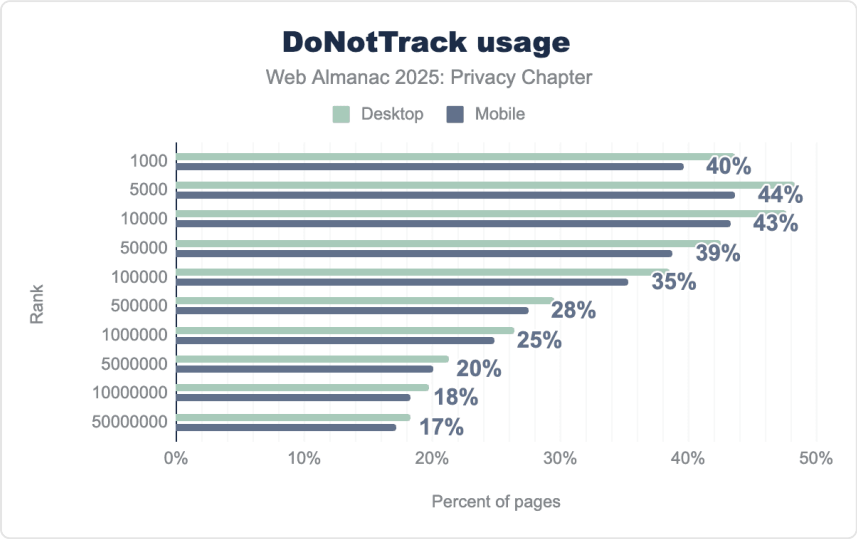


図8.21. Do Not Trackの使用状況

標準としてほぼ廃止され、法的根拠[333]がほとんどなく広告主のほとんどが無視しているにもかかわらず、Do Not Trackシグナルはウェブ全体に残り続けています。興味深いことに、採用率はサイトの人気度と強い相関があります。上位10,000サイトの中でDNT検出は約43%でピークに達し、たとえ実際の影響が疑わしいとしても、ロングテールのサイトはレガシーなプライバシーシグナルを維持する傾向があります。

モバイルの採用率はすべてのランク層でデスクトップをわずかに上回っていますが、差は僅かです。最も急激な低下は上位100,000サイト（35%）と500,000サイト（27%）の層の間で起きており、中規模以下のサイトはDNTをチェックする可能性がはるかに低いことを示しています。これらのサイトが単に検出するだけでなく実際にシグナルを尊重しているかどうかは、DNTへの準拠が一度も強制されてこなかったため、未解決の問題として残っています。

Global Privacy Control

Global Privacy Control（GPC）は、データの販売または共有のオプトアウトのユーザー設定を伝えるブラウザシグナルです。Do Not Trackとは異なり、GPCはCCPA/CPRAの下で法的根拠を持ちます。ウェブサイトはそれを有効なオプトアウトリクエストとして扱わなければなりません。Firefox、Brave、SafariはすでにGPCをサポートしており、2027年までにブラ

333. https://www.loeb.com/en/insights/publications/2013/10/california-enacts-law-requiring-do-not-track-dis_

ウザがこの設定を提供することを義務付けるカリフォルニア州の法律を受けて、Chromeも2026年に実装する予定[334]です。ただし、DNTと同様に、GPCは技術的な水準でウェブサイトが自発的にシグナルを尊重することに依存しています。ブラウザはヘッダー（Sec-GPC: 1）を送信しますが、コンプライアンスを強制することはできません。違いは、GPCを無視すると法的リスクが伴うという点で、これはDNTの純粋に自発的なアプローチよりも効果的であることが証明されるかもしれません。

結論

オンライントラッキングは今日のインターネットにおける標準となっています。実際に、訪問したウェブサイトの75%（デスクトップ）または74%（モバイル）に少なくとも1つのトラッカーが含まれていることが確認されました。

Googleはトラッキングの分野で引き続き支配的な地位を占めており、Facebookがこれに続いています。一見すると、オンライントラッキングはよりターゲットを絞った広告を配信するために活用できる大企業にとって収益性の高い手段です。しかし、少数の集中したプレイヤー間でのトラッキング情報の統合は、よりプライバシーを意識するユーザーにとって懸念事項となっています。

トラッキングを回避する取り組みは絶えず展開され、また回避され続けています。例えば、`medium.com` がバウンスシーケンスで観察されましたが、これは秘密のトラッキングではなく機能的な目的のためと考えられます。一方で、実際のUser-Agent文字列の代わりにUser-Agent Client Hintsを共有するなど、より安全なブラウザポリシーについても考察しました。

オンライントラッキングを管理する法律と規制は、それに準拠するために展開されるメカニズムとともに進化し続けています。TCFの最新バージョン（v2）の不完全な実装と低い採用率が見られますが、IABによる新しい追加であるGlobal Privacy Platformの採用増加も伴っています。さらに、Consent Management Platformの状況にも変化が見られます。

334. *https://chromestatus.com/feature/5137324344213504*

著者

Rumaisa Habib

RumaisaHabib rumaisahabib https://rumaisahabib.com/

Rumaisa Habibは、スタンフォード大学の実証セキュリティ研究グループで研究するPhD課程3年生です。主な研究関心は、大規模なインターネット計測を活用して西洋以外のコンテキストにおけるウェブの理解を深めることです。

Vinod Tiwari

@@securient securient https://blog.securient.io

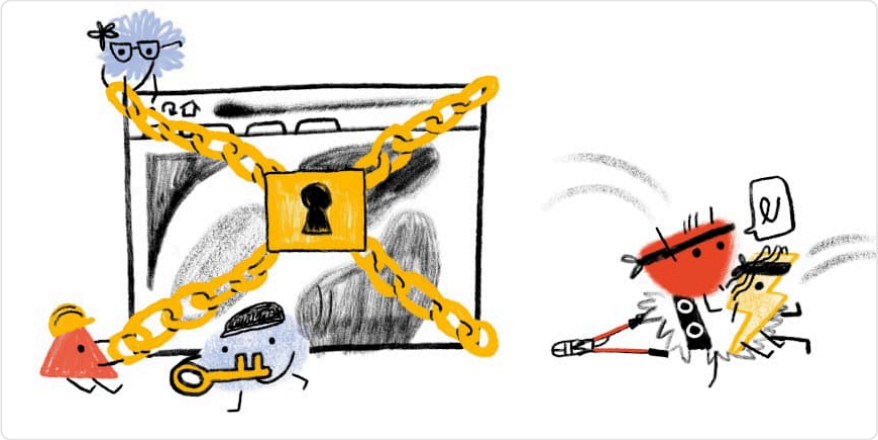
VinodはPIP Labsのスタッフセキュリティエンジニアで、Amazon、Zapier、HackerOneなどの企業でのサイバーセキュリティ経験が10年以上あります。ペネトレーションテストとクラウドセキュリティを専門とし、Mediumでセキュリティについて執筆し、従来型およびWeb3環境の新たな脅威を積極的に研究しています。

Nurullah Demir

nrllh nurullahdemir https://ndemir.com

Nurullah Demirは、スタンフォード大学の訪問ポスドク研究者であり、サイバーセキュリティプラットフォームSecuSeekの創設者です。インターネット規模のセキュリティとプライバシーリスクに焦点を当てた研究を行っています。

部 II 章 9
セキュリティ



Vik Vanderlinden と Gertjan Franken によって書かれた。
Anirudh Duggal、 *Martina Kraus*、 *Gertjan Franken*、 *Brian Clark*、 *Jannis Rautenstrauch と Vinod Tiwari によってレビュ。*
Daan Vansteenhuyse による分析。
Barry Pollard 編集。
Sakae Kotaro によって翻訳された。

はじめに

多くの人々の生活のますます多くの部分がオンラインに移行するにつれて、個人データも同様にオンライン化されており、ウェブセキュリティはかつてないほど重要になっています。日常的に使用する多くのシステムは、データを盗もうとしたり混乱を引き起こそうとする攻撃者にとって依然として魅力的なターゲットとなっています。今年もまた、現代の脅威の規模と複雑さが示されました。DDoS攻撃の規模と頻度は増加し続けており、記録された最大の攻撃は11月に31.4 Tbpsに達しました[335]。サプライチェーンの脆弱性は前例のない規模に拡大し、Shai-Hulud 2.0攻撃[336]は1,000以上のnpmパッケージを侵害し、27,000以上のGitHubリポジトリに感染したと報告されています。また、React2Shell[337]として知られるReactの重大な脆弱性により、開発者はアプリケーションの迅速な更新に追われました。

335. https://blog.cloudflare.com/radar-2025-year-in-review/#hyper-volumetric-ddos-attack-sizes-grew-significantly-throughout-the-year
336. https://www.hackerone.com/blog/shai-hulud-2-npm-worm-supply-chain-attack
337. https://www.microsoft.com/en-us/security/blog/2025/12/15/defending-against-the-cve-2025-55182-react2shell-vulnerability-in-react-server-components/

本チャプターでは、ウェブを保護することを目的とした仕組みと、様々な理由によりそれらが機能しない場合について分析します。Transport Layer Security（TLS）やサードパーティコンテンツインクルージョンに対する保護などのウェブセキュリティの核心要素を探ります。これらのセキュリティ対策の採用がどのように進化しているか、攻撃防止にどのように役立っているか、そして誤設定がどのように適切な機能を妨げるかについて解説します。さらに、セキュリティに関連するいくつかのwell-known URIも分析します。

採用状況の測定だけでなく、場所、技術スタック、ウェブサイトの業種など、セキュリティ機構の採用を促進する要因も探ります。Web Almanacの過去のエディションのデータと比較することで、ウェブセキュリティの状態における長期的なトレンドと変化を把握することができます。

トランスポートセキュリティ

HTTPS[338]はTransport Layer Security（TLS）[339]を使用してクライアントとサーバー間の接続を保護します。年々、HTTPSを使用するウェブサイトが増加し、ユーザーのセキュリティが向上しています。今年、HTTPS経由で送信されるすべてのリクエストの割合は前年比で再び上昇し、モバイル接続で98.8%を超えました。

図9.1. HTTPSを使用するリクエストの割合（モバイル）。

平文のHTTPではなくHTTPSを使用して送信されるリクエストの割合は100%に近づき続けており、2024年版Web Almanac[340]と比較してモバイルでさらに0.74%上昇しました。

338. *https://developer.mozilla.org/docs/Glossary/https*
339. *https://www.cloudflare.com/en-gb/learning/ssl/transport-layer-security-tls/*
340. *https://almanac.httparchive.org/ja/2024/security#fig-1*

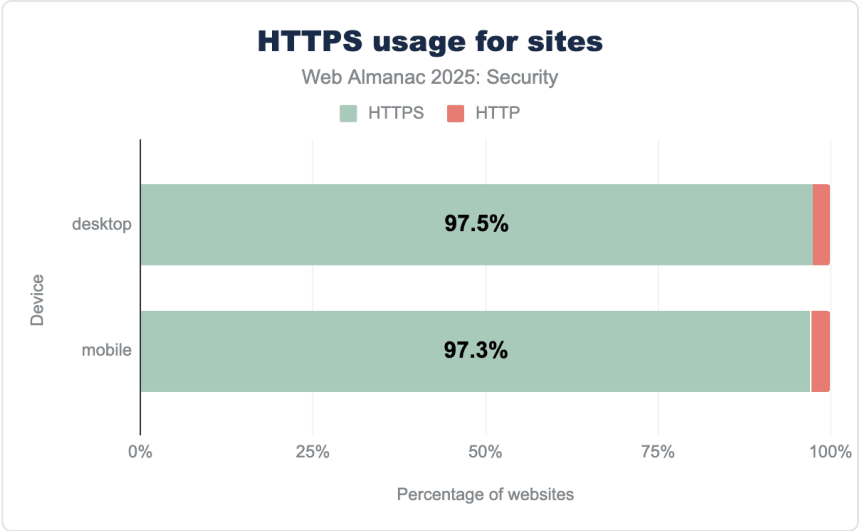


図9.2. HTTPSを使用するホームページの割合。

HTTPS経由で提供されるホームページの数にも前向きな進化が見られます。この数字はモバイルで95.6%から97.3%に上昇しました。多くのウェブサイトが多数の（多くの場合セキュアな）サイトへのサードパーティリクエストを送信するため、この数値はHTTPS経由で送信されるリクエストの割合よりも低くなる傾向がありますが、幸いにも毎年上昇し続けています。

これらの指標で前向きなトレンドが続き、HTTPSを使用するサイトの割合が100%に近づいていることは喜ばしいことです。これらの高い数値の一部は、ブラウザベンダー（Chrome[341]、Firefox[342]、Safari[343]）が平文のHTTPにフォールバックする前にまずHTTPSで通信しようとし、ユーザーにセキュリティ警告を表示することが多いことで、サイトオーナーにTLSの採用を促していることで説明できます。

プロトコルバージョン

ここ数年、TLS1.3[344]は最高のセキュリティを実現するための推奨プロトコルバージョンとなっています。最新バージョンはTLS1.2で欠陥が発見されたアルゴリズムを廃止[345]し、前方秘匿性などのより強固なセキュリティ保証を提供します。QUICも内部的にTLSを使用しており、TLS1.3と同様のセキュリティ保証を提供[346]しています。

341. *https://blog.chromium.org/2021/07/increasing-https-adoption.html#:~:text=Beginning%20in%20M94%2C%20Chrome%20will%20offer%20HTTPS%2DFirst%20Mode*
342. *https://support.mozilla.org/en-US/kb/https-first*
343. *https://webkit.org/blog/16301/webkit-features-in-safari-18-2/#security-and-privacy*
344. *https://www.rfc-editor.org/rfc/rfc8446*
345. *https://www.cloudflare.com/en-in/learning/ssl/why-use-tls-1.3/#:~:text=A%20number%20of%20outdated%20cryptography%20features%20resulted%20in%20vulnerabilities%20or%20enabled%20specific%20kinds%20of%20cyber%20att*
346. *https://community.cloudflare.com/t/how-is-quic-a-direct-comparison-to-tls-1-3-and-tls-1-2/543349/*

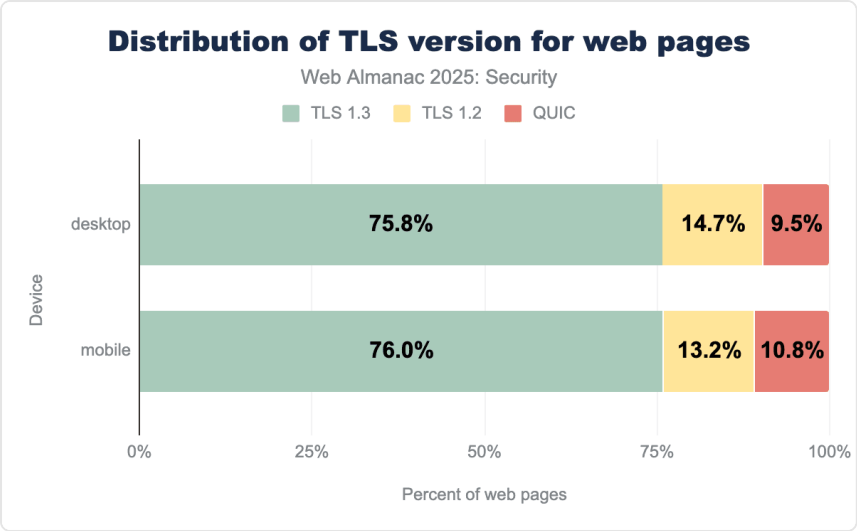


図9.3. 使用中のTLSバージョンの分布。

2024年版Web Almanac[347]と同様に、QUICとTLS1.3の両方で使用が増加していることが分かります。TLS1.2を使用していた一部のサイトがTLS1.3に移行し、TLS1.2またはTLS1.3を使用していた一部のサイトがQUICに移行したと考えられます。TLS1.2は今年さらに4.5%減少しました。今年はQUICの採用がやや鈍化し、QUICを使用するサイトの割合は0.8%しか上昇しませんでした。将来的にはTLS1.2が（長い時間をかけて）徐々に廃止され、QUICが引き続き普及していくと予想されます。

暗号スイート

暗号化されたチャネルでの通信を開始するには、双方が同じ暗号アルゴリズム（または暗号スイート[348]）を使用してお互いを理解する必要があります。そのため、通信前に使用する暗号スイートに合意します。ほとんどのリクエストはGalois Counter Mode（GCM）[349]暗号を使用した接続で行われていることが分かります。これはパディング攻撃[350]への耐性と、TLS1.3で必須とされている[351]関連データ付き認証暗号化（AEAD）[352]の提供により好まれています。また、より安全な256ビット版ではなく、128ビット鍵の使用が前年比4%増加していることも分かります。NISTは128ビット暗号スイートをセキュアとみなしています[353]が、256ビット版はさらに強力なセキュリティ保証を提供します。今年のAlmanacのデータセットに

6#:~:text=TLS%201.2%2C%20TLS%201.3%2C%20and%20QUIC%20share%20similar%20security%20characteristics%20but%20they%20are%20different
347. https://almanac.httparchive.org/ja/2024/security#プロトコルのバージョン
348. https://learn.microsoft.com/en-au/windows/win32/secauthn/cipher-suites-in-schannel
349. https://www.wikipedia.org/wiki/Galois/Counter_Mode
350. https://blog.qualys.com/product-tech/2019/04/22/zombie-poodle-and-goldendoodle-vulnerabilities
351. https://www.rfc-editor.org/rfc/rfc8446#:~:text=Those%20that%0A%20%20%20%20%20%20remain%20are%20all%20Authenticated%20Encryption%20with%20Associated%20Data%0A%20%20%20%20%20%20(AEAD)
352. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=2628d946bda9f3d3b087e5c4846e76ae0fb07b6b
353. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar3.ipd.pdf

は前年よりも多くのリクエストが含まれているため、この変化はデータセットの拡大に起因する可能性があります。使用中の暗号スイートはあまり多様ではなく、実際にデータセットで使用が確認された暗号スイートはわずか5種類です。これはTLS1.3がGCMまたは最新のブロック暗号のみを許可[354]しているためと考えられます。

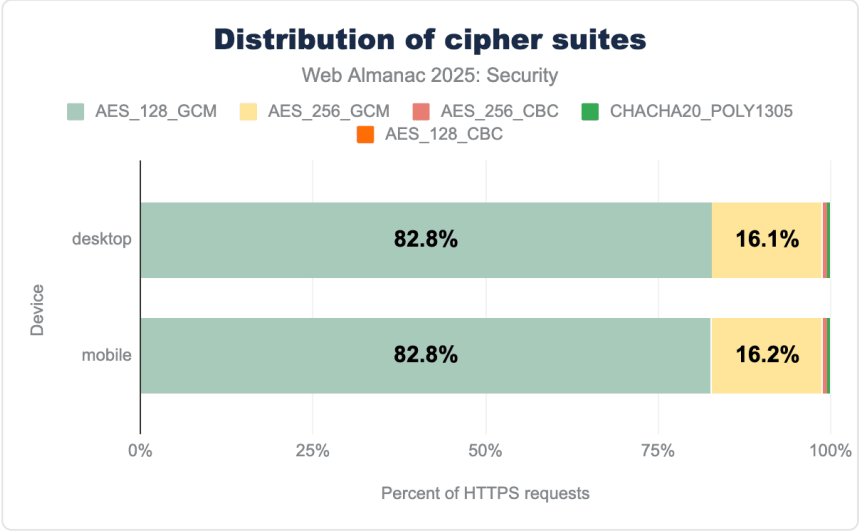


図9.4. 使用中の暗号スイートの分布。

TLS1.3は前方秘匿性[355]をサポートするアルゴリズムのみを許可しています。TLS1.3の高い採用率から、多くのリクエストが前方秘匿性の要件を満たすことが期待されます。

354. https://datatracker.ietf.org/doc/html/rfc8446#page-133
355. https://www.wikipedia.org/wiki/Forward_secrecy

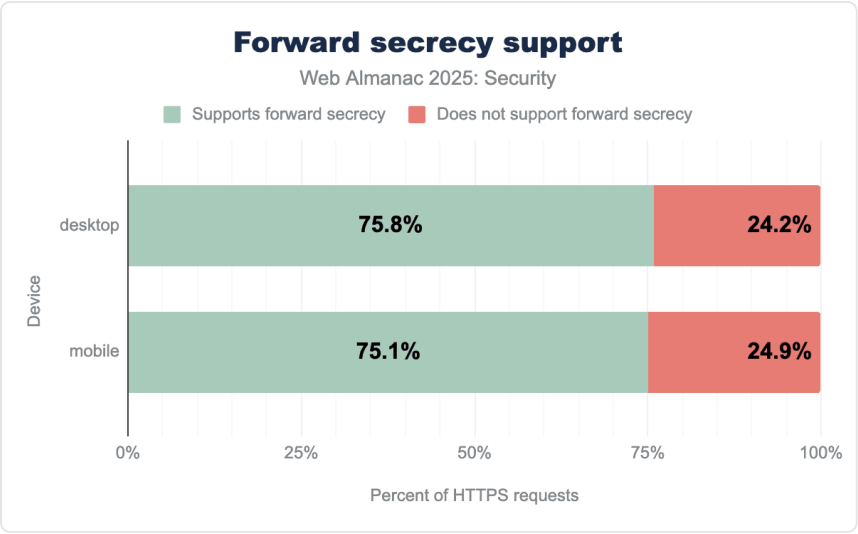


図9.5. 前方秘匿性をサポートするリクエストの割合。

予想に反して、前方秘匿性のある接続で送信されるリクエストの数が比較的少ないことが分かります。2024年版Web Almanac[356]以降、前方秘匿性リクエストは20%減少しています。現在、このメトリクスにはTLS1.2とTLS1.3の前方秘匿性暗号のみが含まれており、QUICのTLSは含まれていないようで、これが確認された低下の説明となる可能性があります。

認証局

TLSを使用するためには、サイトは認証局（CA）[357]から証明書を取得する必要があります。ブラウザが多数のCAを信頼しているため、サイトの証明書はブラウザによって有効な証明書として識別されます。その後、証明書はブラウザとサイトのサーバー間の安全な通信に使用されます。

356. *https://almanac.httparchive.org/ja/2024/security#fig-5*
357. *https://www.ssl.com/faqs/what-is-a-certificate-authority/*

発行者	デスクトップ	モバイル
R11	20.80%	21.86%
R10	20.75%	21.73%
WE1	16.35%	17.24%
E6	4.50%	4.65%
E5	4.31%	4.42%
Sectigo RSA Domain Validation Secure Server CA	3.60%	3.52%
Go Daddy Secure Certificate Authority - G2	1.77%	1.60%
WR1	1.16%	1.40%
Amazon RSA 2048 M02	1.37%	1.33%
Amazon RSA 2048 M03	1.30%	1.25%

図9.6. 発行者ごとの証明書発行割合（上位10件）

前年と比較すると、当時人気だったLet's Encrypt[358]のR3中間証明書が発行者として消えていることが分かります。これは（2025年9月に）期限切れになった[359]ため予想されていたことで、前年からすでに他の中間証明書への移行が始まっていました。R10とR11の中間証明書[360]がR3を引き継ぐ新しい証明書です。現在、中間鍵のピン留めを防ぐことを明示的な目標[361]として、2つの中間RSA証明書（R10とR11）と2つの中間ECDSA証明書（E5とE6）があります。上位5発行者の中でLet's Encrypt以外の唯一の証明書はWE1で、Google Trust Services（GTS）[362]のものです。リストにはGTSのWR1も含まれています。これらの証明書はGTSの新世代中間証明書の一部であり、前年に見られたGTS CA 1P5発行者などが期限切れになっています。

52.6%

図9.7. Let's Encryptが発行したモバイルページの割合。

Let's Encryptの証明書を使用するサイトの総シェアは、前回の56%からわずかに低下して52.6%になりました。データから分かるように、GTSのWE1証明書が発行する証明書のシェアが増加していることが一因として挙げられます。ただし、GTSが発行した証明書（WE1

358. https://letsencrypt.org/
359. https://crt.sh/?id=3334561879
360. https://letsencrypt.org/2024/03/19/new-intermediate-certificates.html
361. https://letsencrypt.org/2024/03/19/new-intermediate-certificates.html#rotating-issuance
362. https://pki.goog/

他）の総シェアは計算されていません。

HTTP Strict Transport Security

HTTP Strict Transport Security[363]は、サーバーがブラウザに対して、まずHTTPを試みてHTTPSへのリダイレクトに従うのではなく、このドメインのページにはHTTPSのみでアクセスするよう指示できるレスポンスヘッダーです。

36%

図9.8. モバイルでHSTSヘッダーを使用するページの割合。

HSTSヘッダーを使用するページ数は引き続き増加しており、前回のエディション[364]と比較して6ポイント上昇し、モバイルで訪問された全ページの36%に達しています。

サーバーはヘッダーにいくつかのディレクティブを含めて、ブラウザに追加の設定を伝えることができます。例えば、`max-age` ディレクティブはブラウザにHTTPSのみを使用し続ける期間を伝えます。他のディレクティブ `includeSubDomains` と `preload` はオプションです。

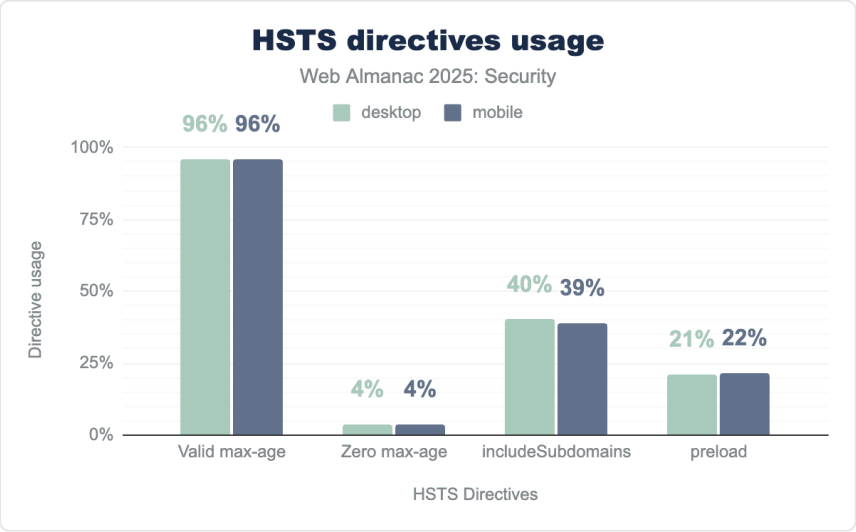


図9.9. 指定されたHSTSディレクティブの使用状況。

363. https://developer.mozilla.org/docs/Web/HTTP/Headers/Strict-Transport-Security
364. https://almanac.httparchive.org/ja/2024/security#fig-8

有効な `max-age` を持つレスポンスのシェアは96%にわずかに増加しました。`includeSubdomains` と `preload` ディレクティブはそれぞれ約4%増加しており、一部のサイトが両方のディレクティブを同時に設定し始めたことを示している可能性があります。非公式[365]の `preload` ディレクティブには `includeSubdirectories` の設定と `max-age` が少なくとも1年の値を持つことが必要です。`preload` ディレクティブを使用することで、サイトはブラウザが初回接続時でもドメインとそのサブドメインに常にHTTPSでアクセスすることを保証できます（preloadなしのHSTSを使用する場合は必ずしもそうとは限りません）。

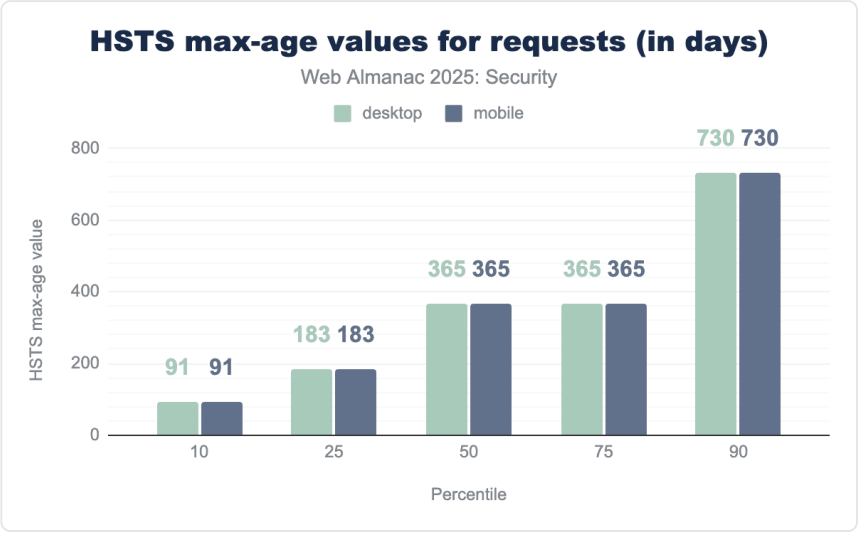


図9.10. パーセンタイル別のHSTSの `max-age` 値の分布。

有効な `max-age` 値の分布は、低いパーセンタイルを除いてほぼ同じままです。10パーセンタイルでは30日から91日へと大きな増加が見られ、非常に低い `max-age` 値を設定するサイトが減少していることを示しています。中央値は1年つまり365日のままです。

クッキー

クッキーはウェブの重要な部分です。クッキーはウェブサイトが複数のステートレスなリクエストにわたって情報を保存できるようにします。サイトのクッキーを保護するために、Cookiesチャプターで詳しく報告されている多くのブラウザ組み込み機能があります。このチャプターでは特にクッキー属性、クッキープレフィックス、永続性（有効期限）のセクションを参照します。

365. https://developer.mozilla.org/docs/Web/HTTP/Headers/Strict-Transport-Security#preloading_strict_transport_security

コンテンツインクルージョン

コンテンツインクルージョンはウェブのコアコンポーネントです。コンテンツデリバリーネットワーク（CDN）からのページ、CSS、JavaScriptや共有ソースからの画像をインクルードできることは、ウェブが構築された基盤の一つです。しかし、これにはサードパーティのコンテンツをインクルードするサイトがそれらのサードパーティリソースに多大な信頼を置くなど、特定のリスクが伴います。もちろん、そのリソースが悪意を持っていないか、悪意のある攻撃者によって侵害されていないという保証はなく、これはサプライチェーン攻撃など多くの深刻な攻撃につながる可能性があります。このリスクを軽減するために、コンテンツインクルージョンを制御するセキュリティポリシーを使用することが重要です。

コンテンツセキュリティポリシー

コンテンツセキュリティポリシー（CSP）[366]は、ウェブサイトがそのページにロードされるコンテンツをきめ細かく制御できるようにします。`Content-Security-Policy` レスポンスヘッダーを設定するか、`<meta>` HTMLタグで定義することで、ウェブサイトは使用中のポリシーをブラウザに伝え、ブラウザはそれを強制します。ポリシーには多くの利用可能なディレクティブがあり、ウェブサイトがどのソースからコンテンツをロードできるかを定義できます。

CSPは特定のリソースのロードをブロックするために使用でき、潜在的なクロスサイトスクリプティング（XSS）攻撃の影響を軽減するのに役立ちます。さらにCSPは、`update-insecure-requests` ディレクティブによる暗号化通信チャネルの使用の強制や、`frame-ancestors` ディレクティブを使用した現在のページがサブリソースとしてロードできるページの制御など、他の目的にも使用できます。これにより、ウェブサイトはクリックジャッキング攻撃から防御できます。

図9.11. 2024年からの `Content-Security-Policy` ヘッダーの採用率の相対的な増加。

CSPの採用は昨年の18.5%[367]から今年の21.9%へと引き続き増加しており、20%近い増加です。Web Almanacの前回版で予測されたように、採用率は20%を超え、引き続き着実に上昇していることを示しています。

366. https://developer.mozilla.org/docs/Web/HTTP/CSP
367. https://almanac.httparchive.org/ja/2024/security#コンテンツセキュリティポリシー

ディレクティブ

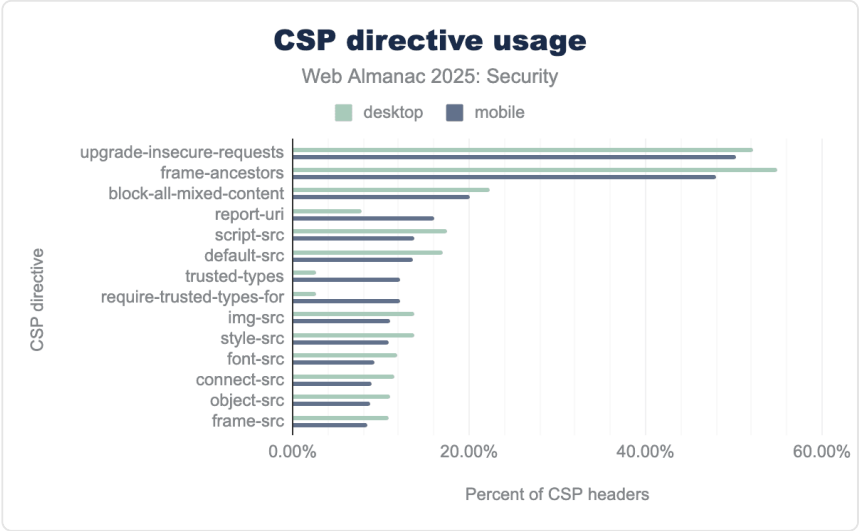


図9.12. CSPで最もよく使用されるディレクティブ。

今年も多くのウェブサイトがCSPを `upgrade-insecure-requests` と `frame-ancestors` ディレクティブのために使用しています。これらのディレクティブを使用するモバイルサイトの相対的なシェアはわずかに減少しましたが、これはスキャンされたCSPヘッダーの数が多いことに起因していると考えられます。絶対数はデスクトップとモバイルでそれぞれ40万と80万のCSPヘッダーが増加しています。

`upgrade-insecure-requests` に置き換えられた `block-all-mixed-content` ディレクティブは、ここ数年と同様に引き続きわずかに減少しています。このディレクティブは非推奨[368]なので、これは良いニュースです。

368. *https://developer.mozilla.org/docs/Web/HTTP/Reference/Headers/Content-Security-Policy/block-all-mixed-content*

ポリシー	デスクトップ	モバイル
`upgrade-insecure-requests`	21.45%	22.79%
`block-all-mixed-content; frame-ancestors 'none'; upgrade-insecure-requests;`	19.92%	18.06%
`require-trusted-types-for 'script';report-uri /checkin/_/AndroidCheckinHttp/csprepor t`	2.67%	12.10%
`frame-ancestors 'self'`	7.55%	6.38%
`upgrade-insecure-requests;`	4.92%	4.30%
`frame-ancestors 'self';`	2.53%	2.29%

図9.13. 最も普及しているCSPヘッダー

昨年上位3位にあったのと同じヘッダー値が今年の上位4位にも登場しています。ただし、今年の3番目に一般的なCSPヘッダーは新しいものです。この変化は、今年は昨年のようにデスクトップ使用率ではなくモバイル使用率で最も一般的なヘッダーをソートしているためです。Android関連の特定の `report-uri` エンドポイントを持つtrusted typesポリシーが3位に現れています。

Trusted types[369]は、インジェクションシンク（`element.innerHTML` など）に渡されるパラメータを制限し、プレーン文字列の代わりに適切に型付けされた値のみが渡されるようにするために使用できます。インジェクションシンクに渡される値を制限することで、多くの潜在的なDOM XSS脆弱性を防ぐことができます。観察されたtrusted typesヘッダーはモバイルページの12%以上に表示されています。この特定のCSPポリシー値の高い使用率については直接的な説明はありません。

5位と6位では、`upgrade-insecure-requests` と `frame-ancestors 'self'` ディレクティブが再び登場しますが、今回は末尾にセミコロンがついています。セミコロンはディレクティブを区切るために使用されますが、定義されているディレクティブが1つだけの場合は廃棄することができ[370]、どちらのヘッダー値も同じ効果を持つ有効なCSPポリシーです。

`script-src` のキーワード

CSPに含まれる最も重要なディレクティブの一つは `script-src` です。このディレクティブを使用することで、ウェブサイトはどのスクリプトがページで実行できるかを制御できま

369. https://w3c.github.io/trusted-types/dist/spec/
370. https://w3c.github.io/webappsec-csp/#grammardef-serialized-policy

す。これは攻撃者が任意のスクリプトを実行するのを妨げることができるため、攻撃者を抑制することができます。`script-src` には複数の潜在的なキーワードがあります。

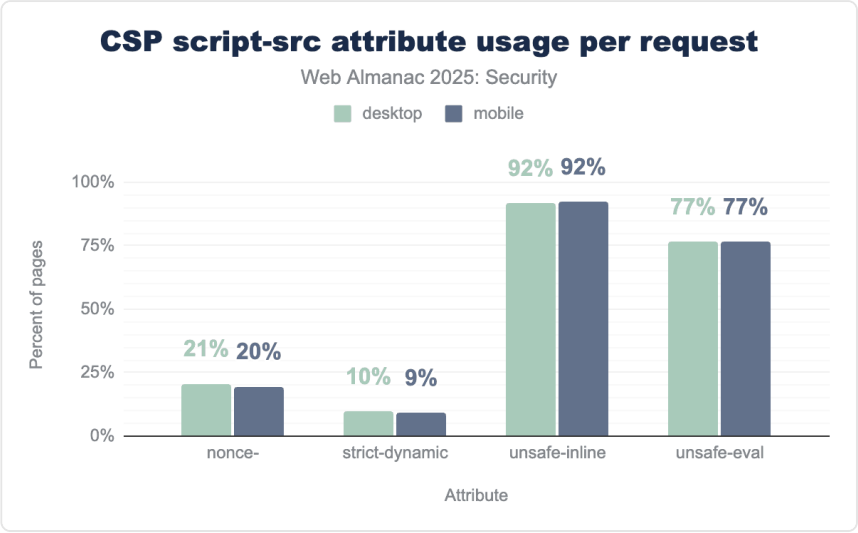


図9.14. リクエストごとのCSP `script-src` キーワードの普及率。

`unsafe-inline` と `unsafe-eval` キーワードが非常に頻繁に使用されていることが分かります。これらのキーワードはそれぞれインラインスクリプトの実行を許可するか、JavaScriptの `eval` 関数の使用を許可するため、CSPの `script-src` のセキュリティへの影響を大幅に低下させます。

昨年[371]と比較して、`script-src` キーワードの使用にほとんど変化がないことが分かります。重要な注意点として、`unsafe-inline` の存在は必ずしもインラインスクリプトが実行できることを意味するわけではありません。場合によっては、CSP仕様[372]に従って `unsafe-inline` は無視されます。例えば、nonceと `strict-dynamic` キーワードがCSPポリシーに追加されている場合がこれに該当します。

371. *https://almanac.httparchive.org/ja/2024/security#script-srcのキーワード*
372. *https://w3c.github.io/webappsec-csp/*

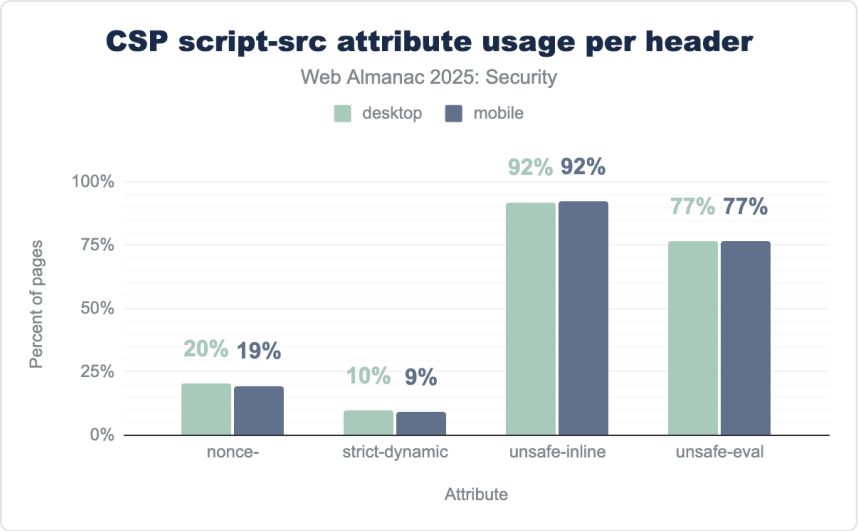


図9.15. ヘッダーごとのCSP `script-src` キーワードの普及率。

ページごとではなくヘッダーごとのキーワード使用状況も確認します。CSPでは、1つのレスポンスに複数のCSPヘッダーが存在し、異なるディレクティブを定義する場合があります。ディレクティブが複数回定義されている場合、ポリシーはブラウザが使用する最も制限的なポリシーを作成するために結合されます[373]。

リクエストごとの値と非常に似た分布が見られ、ほとんどのページがCSPヘッダーを1つだけ使用しているか、設定しているCSPヘッダーの1つだけに `script-src` を使用していることを示しており、ほとんどのページで `script-src` ディレクティブが競合していないことを意味します。

許可されたホスト

CSPは完全に理解して正しく使用するには複雑なセキュリティポリシーです。使用中のCSPヘッダーの長さを確認すると、ポリシーのサイズに大きなバリエーションがあることが分かります。

373. https://content-security-policy.com/examples/multiple-csp-headers/

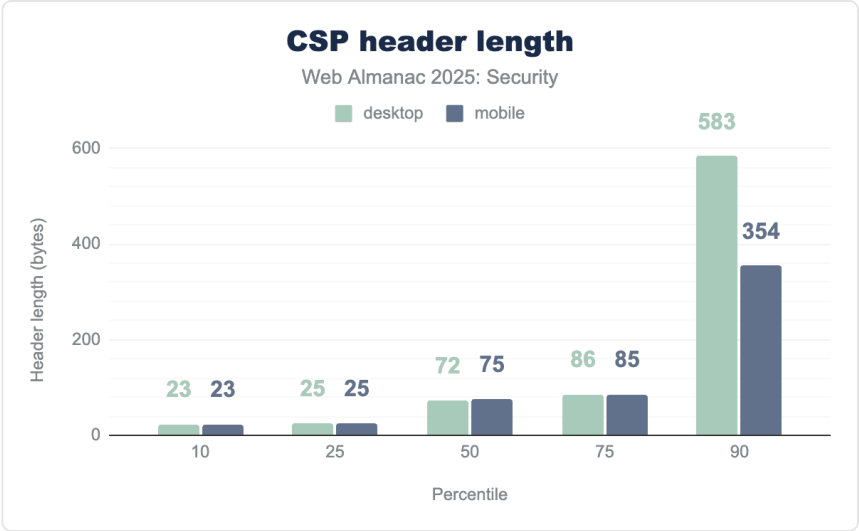


図9.16. CSPヘッダーの長さ。

観察されたすべてのヘッダーのうち、75%が86バイト以下の長さです。これは昨年の75バイト以下よりわずかに長くなっています。90パーセンタイルでより長いポリシーが増加していることが分かります。デスクトップのポリシーは長くなり、モバイルのポリシーはわずかに短くなり、ポリシーの長さの差が広がっています。

ホスト	デスクトップ	モバイル
https://www.googletagmanager.com	0.74%	0.62%
https://fonts.gstatic.com	0.61%	0.49%
https://fonts.googleapis.com	0.60%	0.48%
https://www.google.com	0.54%	0.46%
https://www.google-analytics.com	0.47%	0.38%
https://www.youtube.com	0.41%	0.34%
https://*.google-analytics.com	0.35%	0.31%
https://*.google.com	0.31%	0.30%
https://connect.facebook.net	0.33%	0.29%
https://www.gstatic.com	0.35%	0.27%

図9.17. CSPポリシーで最も頻繁に許可されるHTTP(S)ホスト

CSPヘッダーに含まれる最も一般的にロードされるHTTPSオリジンは、広告、フォント、その他のCDNリソースに使用されるものです。

ホスト	デスクトップ	モバイル
wss://*.hotjar.com	0.18%	0.15%
wss://*.intercom.io	0.07%	0.07%
wss://booth.karakuri.ai	0.08%	0.06%
wss://www.livejournal.com	0.04%	0.05%
wss://*.quora.com	0.03%	0.05%
wss://tsock.us1.twilio.com/v3/wsconnect	0.02%	0.04%
wss://api.smooch.io	0.05%	0.03%
wss://*.zopim.com	0.05%	0.03%
wss://*.pusher.com	0.04%	0.03%
wss://cable.gumroad.com	0.04%	0.03%

図9.18. CSPポリシーで最も頻繁に許可されるWSSホスト

セキュアウェブソケット（`wss://`）オリジンでは、Hotjarが1位を占め、出現数が2倍になっています。他のオリジンは出現数が少ないままです。

Hotjarはウェブサイト分析に使用されており、ウェブサイトの分析情報への継続的な関心を示しており、Intercomはカスタマーサービスに使用されています。また、AIファースト企業として、上位3位に入るkarakuriという日本のAI企業も統計に登場しています。

サブリソース整合性

改ざんされたリソースのロードから保護するために、開発者はサブリソース整合性（SRI）[374]を利用できます。CSPが開発者にリソースがロードされるソースを制限させる一方で、SRIはそれらのリソースが開発者が期待するコンテンツを含むことを確認します。例えば、CDNが侵害され、攻撃者が有効なスクリプトを悪意のあるものに変更することに成功した場合はその逆になります。

`<script>` タグと `<link>` タグで `integrity` 属性を使用することで、開発者はブラウザにリソースの期待するハッシュを伝えることができます。指定されたリソースをロードする際、ブラウザはリソースの内容のハッシュが提供されたハッシュに対応するかどうかを確認

374. *https://developer.mozilla.org/docs/Web/Security/Subresource_Integrity*

し、対応しない場合はリソースのロード/実行を拒否し、ウェブサイトを潜在的に侵害されたコンテンツから保護します。

23.6%

図9.19. SRIを使用するページ（モバイル）。

SRIはデスクトップとモバイルのページでそれぞれ25.9%と23.6%で使用されています。これは昨年[375]から両数字とも約2.5%上昇しており、増加する数の開発者がSRIを使用してページを保護していることを示しています。

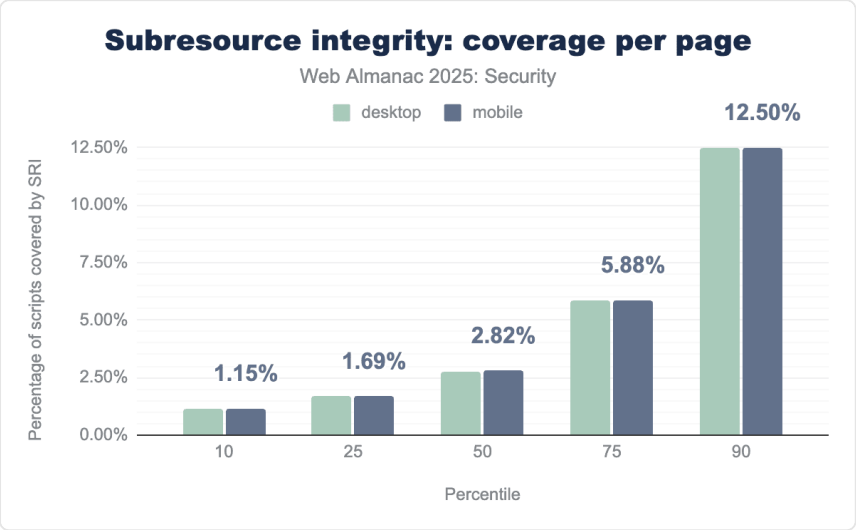


図9.20. ページごとのSRIカバレッジ。

昨年[376]と比較して、ページごとのサブリソースカバレッジの中央値が0.41%低下しています。この低下は、今年のWeb Almanacクローラーがクロールするページ数が多いことによる可能性が高いです。

375. https://almanac.httparchive.org/ja/2024/security#サブリソース完全性
376. https://almanac.httparchive.org/ja/2024/security#fig-25

ホスト	デスクトップ	モバイル
www.gstatic.com	34.70%	34.55%
static.cloudflareinsights.com	8.89%	8.37%
cdnjs.cloudflare.com	7.43%	7.20%
cdn.userway.org	6.10%	6.60%
code.jquery.com	4.96%	4.77%
cdn.shoplineapp.com	4.62%	5.22%
cdn.jsdelivr.net	4.50%	4.06%
d3e54v103j8qbb.cloudfront.net	2.03%	2.30%

図9.21. SRI保護スクリプトが含まれる最も一般的なホスト

SRI保護スクリプトがロードされる最も一般的なホストのリストは、一部のエントリーが相対的な位置を変えながらも、全体的に安定しています。例えば `cloudflareinsights` レコードは相対的な出現率が増加しましたが、2.40%のみです。

昨年と同様に、これらのエントリーのほとんどはCDNであり、SRIでロードおよび保護されるリソースのほとんどがCDNからロードされていることを示しています。これらのスクリプトは開発者の直接制御下にないことが多いため、これは理にかなっています。CDNは多くの場合、多くの開発者が含めることができる大量のスクリプトをホストします。これは自分のサーバーへの負荷が下がるため開発者にとって有益ですが、クライアントにとってもそれを含むすべてのウェブサイトで同じスクリプトを一度ずつロードするのではなく、CDNからスクリプトをキャッシュできるという点で有益です。

自分で制御していないサーバーにホストされているスクリプトを含めるリスクは、開発者として完全に制御している場合よりも高くなります。この外部スクリプトの開発者が悪意のあるコンテンツを含めることを決めるか、サーバーが侵害されてスクリプトが悪意のあるものに置き換えられた場合、SRIなしでは開発者のユーザーがこの悪意のあるコンテンツをロードして実行することになります。ユーザーをこれらの予期しない変更から保護し、どのスクリプトが実行を許可されているかを確実に知るための良い方法がSRIの使用です。

パーミッションポリシー

パーミッションポリシー[377]（以前はFeature Policy）は、カメラ、マイク、センサー（加速度

377. *https://developer.mozilla.org/docs/Web/HTTP/Permissions_Policy*

計など）、位置情報データなど、ブラウザでの特定機能の使用を許可または拒否できるようにするポリシーです。`Permissions-Policy` レスポンスヘッダーを通じて、開発者はメインページとその埋め込みコンテンツによる特定の機能使用を許可または拒否できます。1つの埋め込みリソースに対する特定のポリシーは、`<iframe>` 要素の `allow` 属性を通じて設定できます。

図9.22. 2024年からの `Permissions-Policy` ヘッダーの採用率の相対的な増加（モバイル）。

昨年と比較して[378]、`Permissions-Policy` の使用は相対的に60%近く増加しました。これは大きな相対的な増加ですが、`Permissions-Policy` を使用するウェブサイトの絶対的な割合は、モバイルで3.7%と依然として比較的少ない状況です。採用率の上昇はポリシーにとって良い兆しです。

378. https://almanac.httparchive.org/ja/2024/security#パーミッションポリシー

ヘッダー
`interest-cohort=()`
`accelerometer=(), autoplay=(), camera=(), encrypted-media=(), fullscreen=(), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), midi=(), payment=( picture-in-picture=(), publickey-credentials-get=(), screen-wake-lock=(), sync-x usb=()`
`geolocation=(),midi=(),sync-xhr=(),microphone=(),camera=(),magnetometer=(),gyroscope=(),fullscreen=(self),pa`
`accelerometer=(), autoplay=(), camera=(), cross-origin-isolated=(), display-capture=(self), encrypted-media=(), fullscreen=*, geolocation=(self), gyroscope= keyboard-map=(), magnetometer=(), microphone=(), midi=(), payment=*, picture-in- picture=*, publickey-credentials-get=(), screen-wake-lock=(), sync-xhr=*, usb=() spatial-tracking=(), gamepad=(), serial=()`
`geolocation=(self)`
`accelerometer=(), camera=(), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), payment=(), usb=()`
`accelerometer=(self), autoplay=(self), camera=(self), encrypted-media=(self), fullscreen=(self), geolocation=(self), gyroscope=(self), magnetometer=(self), microphone=(self), midi=(self), payment=(self), usb=(self)`
`browsing-topics=()`
`geolocation=(), microphone=(), camera=()`
`geolocation=self`

図9.23. 最も普及している `Permission-Policy` ヘッダー

上位10位の `Permissions-Policy` 値を確認すると、GoogleのFederated Learning of Cohorts（FLoC）[379]をオプトアウトするためにヘッダーを使用する開発者が少なくなっており、`Permissions-Policy` ヘッダーの11.5%のみが `interest-cohort=()` 値を含んでいます。また、モバイルでの `Permissions-Policy` ヘッダーの10%がこの値を含む、一度に多くの機能をオプトアウトする値が人気の値になっていることも分かります。これは異なる

379. *https://privacysandbox.com/intl/en_us/proposals/floc/*

Permission PolicyでのTopics APIに置き換えられたため[380]と考えられます。また、Googleがこれらの Privacy Sandbox APIを廃止している[381]ため、将来的にさらに減少する可能性が高いです。

上位10位で観察された他のすべての値は、ウェブページと埋め込みコンテンツの権限を制限することを目的としています。パーミッションポリシーはデフォルトでオープンであり、機能の使用を制限するためにはヘッダーで明示的に言及する必要があります。昨年と同様に、デスクトップの `Permissions-Policy` ヘッダーの0.27%が `*` ワイルドカード値を設定しており、`allow` 属性により厳格なポリシーを定義していないページと埋め込みコンテンツにすべての権限を明示的に付与しています。モバイルでは、ワイルドカード値はまったく見つかりませんでした。

前述のように、Permissions PolicはHTML `<iframe>` の `allow` 属性でも定義できます。例えば、埋め込みコンテンツにカメラとマイクへのアクセスを許可するiframeは次のようになります：

```
<iframe src="https://example.com/" allow="camera 'self'; microphone
'self'">
```

モバイルの合計3,330万のiframeのうち、29.2%が `allow` 属性を含んでいることが観察されました。`allow` 属性の使用の相対的な減少は、今年のクロールで前年よりも多くのページが含まれていたため、1,000万以上多くのiframe要素が見つかったことに起因していると考えられます。

380. *https://privacysandbox.google.com/private-advertising/topics/web/controls#opt_out_as_a_developer*
381. *https://privacysandbox.google.com/blog/update-on-plans-for-privacy-sandbox-technologies*

ディレクティブ	デスクトップ	モバイル
encrypted-media	60.90%	70.66%
autoplay	34.99%	41.83%
picture-in-picture	26.10%	36.63%
gyroscope	23.76%	34.20%
accelerometer	23.76%	34.20%
clipboard-write	20.39%	26.24%
join-ad-interest-group	24.23%	17.35%
web-share	12.07%	15.67%
attribution-reporting	3.93%	2.35%
run-ad-auction	3.84%	2.26%

図9.24. 最も普及している`allow`属性ディレクティブ

興味深いことに、昨年の上位3つの`allow`属性値（`join-ad-interest-group`、`attribution-reporting`、`run-ad-auction`）は昨年と比べて極めて低い採用率になっています。これらの値をiframe要素に追加したと仮定されていた大手プレイヤーが、その変更を元に戻した可能性があります。昨年の上位10位に入るその他の値は、`allow`属性値への組み込みが全体的に大幅に増加しており、絶対的な変化は最大+50%に達しています。

iframeサンドボックス

`<iframe>`要素に`sandbox`属性を使用することで、開発者は`<iframe>`に埋め込まれたリソースが侵害されたり悪意を持ったりした場合のいくつかの攻撃からユーザーを保護できます。`sandbox`属性の値では、開発者が`<iframe>`にロードして表示されるコンテンツに対してどのような制限を設けるべきかを指定できます。例えば、以下の`<iframe>`は埋め込みウェブページがスクリプトを実行できるようにします：

```
<iframe src="https://example.com/" sandbox="allow-
scripts"></iframe>
```

sandbox属性の使用は2024年版から20.0%から22.7%に上昇しており、埋め込みコンテンツ

による潜在的な悪用からユーザーを保護したいと考える開発者が増加していることを示しています。

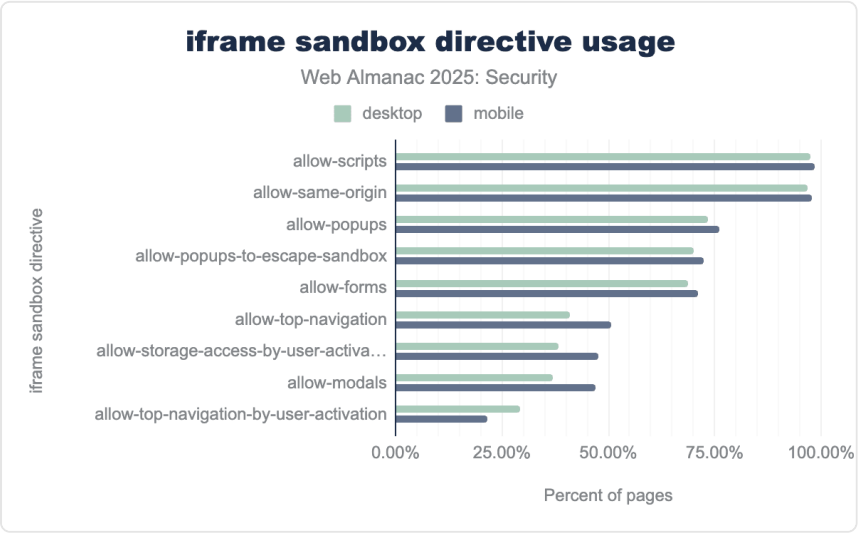


図9.25. iframeでのsandboxディレクティブの普及率。

昨年[382]と同様に、`sandbox`属性を持つすべてのiframeのうち、98.5%（モバイル）が`allow-scripts`ディレクティブを含めることで埋め込みウェブページがスクリプトを実行できるようにするために使用していることが分かります。2位は、モバイルiframeの97.8%で使用されており、開発者がロードされたリソースを埋め込み元のオリジンの一部にするための`allow-same-origin`です。

新興のセキュリティヘッダー

TLS、HSTS、CSPポリシー、SRI、iframeの`sandbox`属性などの既存のセキュリティ機能のほとんどで上昇が見られます。これらの既存機能に加えて、時代が進むにつれて新しい攻撃が作成され、より多くの防御が実装されています。このセクションでは、実際の環境で見られる頻度が増えている比較的新しいドキュメントポリシーについて説明します。

ドキュメントポリシー

ドキュメントポリシー[383]は、2022年に最終更新されたコミュニティグループレポートの草案

382. https://almanac.httparchive.org/ja/2024/security#iframeサンドボックス
383. https://wicg.github.io/document-policy/

です。もともとは、パーミッションポリシーモデルに合わなかったり、複雑さを過度に増加させたりするパーミッションポリシーへの提案された追加に対する応答として作成されました。

ドキュメントポリシーはパーミッションポリシー、CSP、サンドボックスなどの関連メカニズムと比べていくつかの利点があります。パーミッションポリシーよりもきめ細かく、異なる継承モデルを持っており：子リソースは互換性がある場合は親が選んだ特定のポリシーを上書きできます。CSPよりも一般的で、リソースがロードされた後の権限に関するディレクティブを持っており、リソースをロードできるオリジンを決定するだけではありません。サンドボックスより拡張が容易なのは、サンドボックスで言及されていない機能はデフォルトでブロックされており、新しいものを追加することが非常に困難だからです。

ヘッダー	デスクトップ	モバイル
`include-js-call-stacks-in-crash-reports`	68.48%	71.99%
`js-profiling`	12.66%	15.24%
`js-profiling; include-js-call-stacks-in-crash-reports`	17.41%	11.94%
`force-load-at-top`	1.25%	0.65%
`no-font-display-late-swap`	0.06%	0.05%

図9.26. 最も一般的なdocument policyヘッダー値

使用中のドキュメントポリシーヘッダーのうち、3分の2以上がクラッシュレポートにコールスタックを含めるために使用されていることが分かります。`js-profiling` ディレクティブと組み合わせると、これら2つの機能が現在のユースケースの大多数を占めています。現在、合計で19の異なるディレクティブを含むポリシー値が見つかります。一般的には定義されているディレクティブがより多い場合もありますが、現時点ではその合計数は把握できていません。

現在、デスクトップとモバイルでそれぞれ約24,000と29,500ページしか見つかっておらず、訪問した総ページ数の0.10%です。ドキュメントポリシーヘッダーの採用率は今後も上昇すると予想されますが、将来的な採用は急速には進まないかもしれません。

攻撃防御

ブラウザが実装しているウェブサイトへの多くの防御がある一方で、すべての可能性とベス

トプラクティスの概要を把握することは困難です。さらに、保護がオプトインであり、デフォルトで有効になっていない場合は、さらなる課題となります。開発者は最新の攻撃と、それらの攻撃からユーザーを保護するために存在する防御について常に最新情報を把握しておく必要があります。このセクションでは、ウェブ全体でどの攻撃防御手段が使用されているかを評価します。

セキュリティヘッダーの採用

多くの保護メカニズムがHTTPレスポンスヘッダーを通じて設定できます。これらのヘッダーの値に基づいてブラウザはこれらの保護を強制します。すべてのセキュリティメカニズムがすべてのウェブサイトに関連するわけではありませんが、セキュリティヘッダーがまったくない場合はセキュリティに対する緊急性の欠如を示す可能性があります。

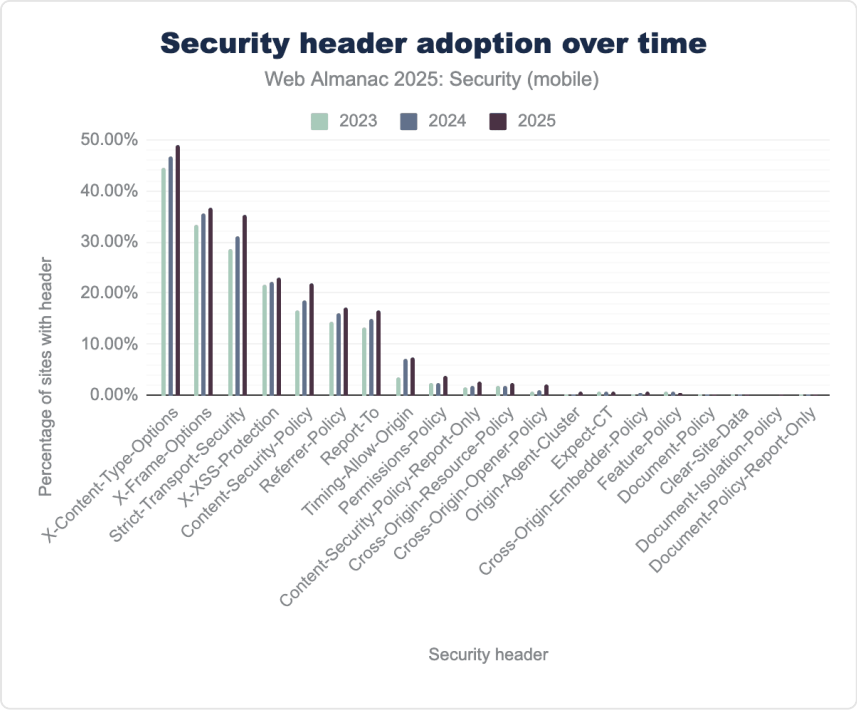


図9.27. 経時的なモバイルページでのサイトリクエストのセキュリティヘッダー採用率。

昨年と同様に[384]、採用が減少しているヘッダーはわずかです。`Feature-Policy` ヘッダーは `Permissions-Policy` に取って代わられたため廃止されており、採用が減少しているこ

384. *https://almanac.httparchive.org/ja/2024/security#セキュリティヘッダーの採用*

とは驚くことではありません。他の2つ（`Clear-Site-Data` と `Document-Policy-Report-Only`）の採用率は非常に低く（それぞれ0.01%と0.00001%）、採用率の相対的な変化は大きく見えることがありますが、絶対的な差は実際には小さいです。これは全体的なセキュリティヘッダーの採用が時間とともに増加し続けていることを意味し、ウェブセキュリティ全体にとって前向きな兆候です。

2024年版からの最も強い上昇は `Strict-Transport-Security`（+4.02%）、`Content-Security-Policy`（+3.39%）、`X-Content-Type-Options`（+2.30%）です。

`Origin-Agent-Cluster`

`Origin-Agent-Cluster` は、正しく設定されている場合、ドキュメントに使用されるリソース（オペレーティングシステムプロセスなど）を同じオリジンのドキュメントと共有するリクエストをブラウザに伝えます。ブラウザはリクエストに応じることも応じないこともあり、クライアントはJavaScriptを使用してリクエストが実際に応じられたかどうかを確認できます。

使用率は依然として低く、モバイルサイトの0.47%、デスクトップサイトの0.38%がこれを使用していますが、彼らが何のために使用しているかを詳しく見てみましょう：

ヘッダー	デスクトップ	モバイル
`?1`	74.11%	90.32%
`?0`	25.79%	9.60%
`1`	0.08%	0.07%
`0`	0.01%	0.01%

図9.28. 最も一般的な `Origin-Agent-Cluster` ヘッダー値

ブール値はHTML Living Standardで[385]疑問符で始まる文字列として定義されています。これは `1` や `0` のような値がこのヘッダーに対して無効であることを意味します。幸いにも、これらの値の使用は限定的です。`?1` 値は `Origin-Agent-Cluster` の唯一の有効値であり、開発者がこの機能にオプトインしたいことを伝えるために使用され、他のすべての値は無視されます。モバイルでは、ヘッダーの90%以上が有効な `?1` 値を持っています。残念ながら、ヘッダー値の0.07%は `1` であり、開発者が専用リソースの使用をリクエストしたいにもかかわらず無視される値です。

385. *https://httpwg.org/specs/rfc8941.html#boolean*

`document.domain`の使用

`document.domain`を使用することで、開発者は現在のドキュメントのドメイン部分を読み取ることができ、新しいドメインを設定することもできます（現在のドメインのスーパードメインのみ許可）。その後、ブラウザは同一オリジンポリシーのチェックに新しいドメインをオリジンとして使用します。ただし、このプロパティの使用は現在非推奨となっており、ブラウザはまもなくこのプロパティのサポートを停止する可能性があります。

Blink機能	デスクトップ	モバイル
DocumentSetDomain	0.49%	0.36%
DocumentDomainSetWithNonDefaultPort	0.16%	0.14%
DocumentDomainEnabledCrossOriginAccess	0.0008%	0.0004%
DocumentDomainBlockedCrossOriginAccess	0.0002%	0.0001%
DocumentOpenAliasedOriginDocumentDomain	0.00008%	0.00001%

図9.29. 特定のBlink機能に基づく`document.domain`の使用

デスクトップとモバイルのウェブサイトのうち0.5%未満が`document.domain`セッターを使用してページのオリジンを変更しており、非デフォルトポートで行うサイトはさらに少ないことが分かります。これは前向きなトレンドですが、依然としてコードを更新すべき数万のウェブサイトが存在します。

CSPとX-Frame-Optionsによるクリックジャッキング防御

前述のように、コンテンツセキュリティポリシー（CSP）は`frame-ancestors`ディレクティブを使用してクリックジャッキング[386]攻撃に対して有効です。一部の上位CSPヘッダー値には`'none'`または`'self'`値を持つ`frame-ancestors`ディレクティブが含まれており、ページの埋め込みを全体的にブロックするか、同一オリジンのページへの埋め込みを制限します。

クリックジャッキング攻撃に対する別の防御方法は、`X-Frame-Options`（XFO）ヘッダーを通じてです。XFOヘッダーを設定することで、開発者はドキュメントが他のドキュメントに埋め込めないこと（'DENY'）、または同一オリジンのドキュメントのみに埋め込めること（`SAMEORIGIN`）を伝えることができます。

386. *https://owasp.org/www-community/attacks/Clickjacking*

ヘッダー	デスクトップ	モバイル
SAMEORIGIN	72.48%	72.13%
DENY	24.40%	24.64%
ALLOWALL	0.68%	0.72%
SAMEORIGIN, SAMEORIGIN	0.27%	0.28%
allow-from https://s.salla.sa	0.16%	0.28%

図9.30. 最も普及している`X-Frame-Options`ヘッダー値

今年のデータと2024年[387]のデータではほとんど変化がないことが分かります。`X-Frame-Options`ヘッダーは主に同一オリジンのウェブサイトがページを埋め込めるようにするために使用されています（72.1%）。次に、任意のページが自身のコンテンツを埋め込むことを拒否するために使用されています（24.6%）。

観察される他の値の例はテーブルの3行目から5行目に示されています。`ALLOW-FROM`値はかつて有効でしたが、現在は非推奨でブラウザに無視されます。XFOで`ALLOW-FROM`を使用する代わりに、開発者はCSPの`frame-ancestors`ディレクティブの使用に切り替えるべきです。これらの仕様外の値はあまり見られませんが、ヘッダーを設定することで保護が有効になると期待する開発者によって設定されている場合があります。

クロスオリジンポリシーを使用した攻撃防御

SpectreとMeltdown[388]のようなマイクロアーキテクチャサイドチャネル攻撃とCross-Site Leaks（XS-Leaks）[389]の登場により、クロスオリジンリソースの使用と埋め込みに関するセキュリティの視点が変わりました。これらの脅威に対応して、他のウェブサイトでのリソースのレンダリングを制御し、これらの新しい脅威から保護するための新しいメカニズムが作成されました。

クロスオリジンポリシーとして知られる複数の新しいセキュリティヘッダーがこれらの課題への対応として作成されました：Cross-Origin-Resource-Policy（CORP）、Cross-Origin-Embedder-Policy（COEP）、Cross-Origin-Opener-Policy（COOP）。これらのヘッダーは、開発者がリソースが異なるオリジン間でどのように埋め込まれるかを制御できるようにすることで、サイドチャネル攻撃から保護するメカニズムを提供します。これらのヘッダーのすべての採用が年々増加し続けており、CORPとCOOPの両方が今年2%以上の採用率に達していることが観察されます。

387. *https://almanac.httparchive.org/ja/2024/security#セキュリティヘッダーの採用*
388. *https://spectreattack.com/*
389. *https://xsleaks.dev/*

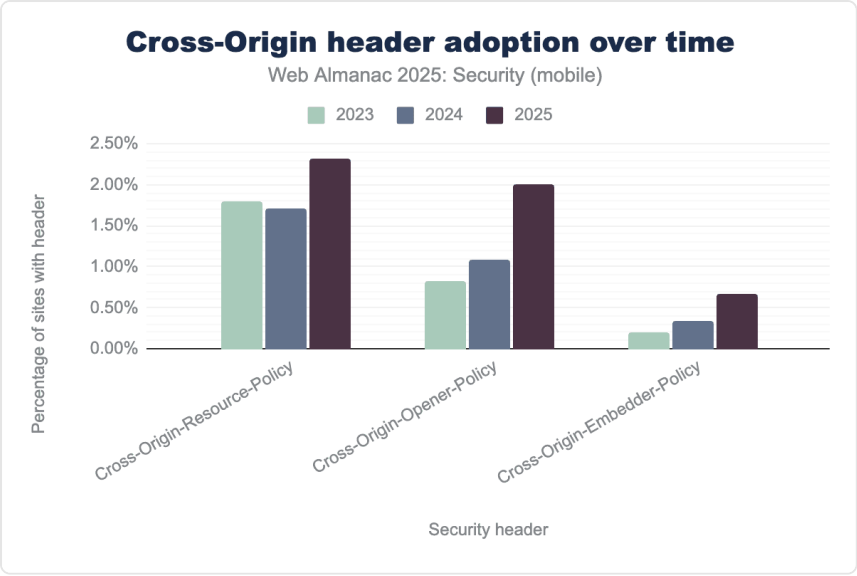


図9.31. クロスオリジンヘッダーの経時的な使用状況。

クロスオリジンエンベッダーポリシー

Cross-Origin-Embedder-Policy（COEP）[390]は、開発者がどのクロスオリジンリソースが現在のドキュメントに埋め込めるかを設定できるようにします。デフォルト（ヘッダーが存在しない場合）では、すべてのクロスオリジンリソースがページに埋め込めます。これはヘッダーが `unsafe-none` 値で設定されている場合と同じ動作です。

COEP値	デスクトップ	モバイル
`unsafe-none`	83.26%	86.52%
`require-corp`	6.68%	4.92%
`credentialless`	2.59%	1.89%

図9.32. 許可された値の有効なバリエーションを含むCOEPヘッダーの普及率

昨年と比較して[391]、ほとんどの開発者は依然として `unsafe-none` 値を使用してすべてのコンテンツが現在のドキュメントに埋め込まれることを明示的に許可するためにCOEPヘッダーを設定しています。この値の使用率はモバイルで依然として86%以上ありますが、昨年か

390. *https://developer.mozilla.org/docs/Web/HTTP/Headers/Cross-Origin-Embedder-Policy*
391. *https://almanac.httparchive.org/ja/2024/security#クロスオリジンエンベッダーポリシー*

ら約2%低下しており、開発者がヘッダーの使用を変更し始めていることを示している可能性があります。

他の値 `require-corp` と `credentialless` は昨年からそれぞれ0.2%と0.3%のわずかな採用増加を見せました。`require-corp` を使用する場合、ブラウザは同一オリジンコンテンツまたはCORPによって埋め込みが許可されたクロスオリジンコンテンツのみがページに埋め込まれることを強制します。

`credentialless` の場合、ブラウザはコンテンツのCORSポリシーに関係なく `no-cors` モードでクロスオリジンリクエストを許可しますが、クッキーはリクエストに添付されません。

クロスオリジンリソースポリシー

COEPと関連して、Cross-Origin-Resource-Policy（CORP）[392]は、現在のドキュメントにどのコンテンツが埋め込まれるかを強制するのではなく、現在のコンテンツがどのドキュメントからアクセスできるかを管理します。

可能な値は `cross-origin`、`same-origin`、`same-site` の3つだけです。`cross-origin` 値はすべてのドキュメントがリソースにアクセスできるようにし、`same-origin` と `same-site` 値はそれぞれ同一オリジンまたはサイトのドキュメントのみにリソースへのアクセスを制限します。開発者はオリジン（スキーム、ホスト、ポート）とサイト（登録可能なドメイン）の違い[393]を認識すべきです。ヘッダーが存在する場合、`no-cors` モードのリクエストはブラウザによってブロックされます。

CORP値	デスクトップ	モバイル
cross-origin	81.36%	80.52%
same-origin	14.40%	15.63%
same-site	3.80%	3.48%

図9.33. CORP ヘッダー値の普及率

ほとんどの場合、ヘッダーはあらゆるクロスオリジンリソースへのアクセスを許可するために使用されています。今年この数字に大きな変化が見られ、昨年[394]から10%以上低下してモバイルで80.5%になっています。一方、`same-origin` 値の使用は約10%上昇しており、開発者がクロスオリジンアクセスからリソースを保護する方向に移行していることを示しています。`same-site` の使用シェアはほぼ同じで、半パーセント未満のわずかな減少を示して

392. https://developer.mozilla.org/docs/Web/HTTP/Cross-Origin_Resource_Policy
393. https://web.dev/articles/url-parts
394. https://almanac.httparchive.org/ja/2024/security#クロスオリジンリソースポリシー

います。

クロスオリジンオープナーポリシー

最後のクロスオリジンポリシーヘッダー、Cross-Origin-Opener-Policy（COOP）[395]は、開発者が `window.open` などのブラウザAPIを通じてページを開いたときに他のページがそのページを参照できる方法を制御できるようにします。

デフォルト値の `unsafe-none` はCOOP保護を無効にできるようにし、これはヘッダーが存在しない場合にも起こります。開発者が `window.open` を使用して `unsafe-none` を使用するページを開いた場合、返された値を使用して開かれたページの特定のプロパティにアクセスでき、これがCross-Site Leaksにつながる可能性があります。

`same-origin` がオープナーと開かれたリソースの両方に存在する場合、`windows.open` によって返された参照はオープナーが使用できます。`same-origin-allow-popups` は、ドキュメントが依然として動作する参照へのアクセスを維持しながら、`unsafe-none` を持つ別のドキュメントを開けるようにします。

最後に、`noopener-allow-popups` は、開かれたドキュメントも同じCOOP値が設定されている場合を除いて、参照に決してアクセスできないようにします。

COOP値	デスクトップ	モバイル
same-origin	58.22%	61.64%
unsafe-none	28.47%	26.82%
same-origin-allow-popups	11.36%	9.95%
noopener-allow-popups	0.03%	0.03%

図9.34. 許可された値の有効なバリエーションを含むCOOPヘッダーの普及率

COOPの最も厳格な `same-origin` 値の使用はモバイルで47.5%から61.6%に引き続き上昇しています。`noopener-allow-popups` は非常に新しい値で、昨年[396]にはまだ存在しませんでした。今年はこの値を使用する少数の採用者が確認されます。`unsafe-none` の使用は10%以上低下しました。これらの変化はCOOPの使用において前向きな進化を示しています。

395. https://developer.mozilla.org/docs/Web/HTTP/Headers/Cross-Origin-Opener-Policy
396. https://almanac.httparchive.org/ja/2024/security#クロスオリジンオープナーポリシー

クロスオリジンアイソレーション

`SharedArrayBuffer` や `Performance.now` などの特定の機密APIにアクセスするためには、サイトがクロスオリジンアイソレーション状態[397]である必要があります。クロスオリジンアイソレーション状態であるためには、開発者がCOEPを `same-origin` に設定し、CORPを `require-corp` または `credentialless` のいずれかに設定する必要があります。その後、ブラウザはこれらのAPIへのアクセスを再び許可します。これはXS Leaksに対する保護を強化します。最近では、開発者はドキュメントアイソレーションポリシー[398]を使用してクロスオリジンアイソレーションにオプトインすることもできます。

`Clear-Site-Data` を使用した攻撃防御

`Clear-Site-Data` HTTPレスポンスヘッダーを使用することで、開発者はクライアントにブラウジングデータをクリアするよう指示できます。ヘッダーの値はクリアするデータの種類を指定します。これはユーザーがウェブサイトからログアウトするときに役立ち、開発者は認証クッキーが確実にクリアされることを確認できます。

`Clear-Site-Data` の採用率を正確に推定することは難しいです。その使用はユーザーをログアウトさせる際に最も価値があることが多いためです。使用しているクローラーはウェブサイトにログインしないため、ログアウト後にヘッダーを使用するサイト数を確認するためにログアウトすることもできません。現時点では、`Clear-Site-Data` ヘッダーを使用している2,024のモバイルホストが確認されており、これはクロールされたホストの総数の0.01%にすぎません。

397. *https://developer.mozilla.org/docs/Web/API/Window/crossOriginIsolated*
398. *https://wicg.github.io/document-isolation-policy/*

クリアサイトデータ値	デスクトップ	モバイル
`cache`	30.82%	29.25%
`*`	17.74%	20.61%
`cookies`	7.02%	8.16%
`"cache"`	8.94%	7.19%
`"storages"`	5.66%	6.08%
`cache, cookies, storage`	2.12%	3.23%
`"cache", "cookies", "storage", "executionContexts"`	2.32%	2.51%
`"cookies"`	2.78%	2.46%
`"storage"`	2.27%	2.03%
`"cache", "storage", "executionContexts"`	1.36%	1.54%

図9.35. `Clear-Site-Data` ヘッダーの普及率

データは、ほとんどの開発者がキャッシュをクリアするために `Clear-Site-Data` ヘッダーを使用しようとしていることを示しています。最も普及している値は `cache` で、次にワイルドカード文字 `*` と `cookies` が続きます。上位3つの値はすべて仕様によると無効です。このヘッダーの値は「引用符付き文字列」でなければならず、つまり `"cache"`、`"*"`、`"cookies"` が同等の有効な値です。上位3つの値を合わせると、モバイルの現在のヘッダー値の58.01%をすでに占めているため、これは懸念されます。

これらの数字は年々かなり変動しますが、ヘッダーの採用率が低いため、非常に少数のホストが相対的な割合を大幅に変更できることで説明できると考えられます。

`<meta>` を使用した攻撃防御

レスポンスヘッダーで設定することに加えて、ウェブのセキュリティメカニズムの一部は `<meta>` タグを使用してHTMLドキュメント内で直接設定できます。その例として `Content-Security-Policy` と `Referrer-Policy` があります。これらのメカニズムへのmetaタグの使用は昨年[399]からほぼ安定しており、CSPとReferrer-Policyでそれぞれ約0.60%と

399. *https://almanac.httparchive.org/ja/2024/security#meta を使用した攻撃の防止*

約2.50%です。昨年と同様に、CSPのわずかな減少とReferrer-Policyのわずかな増加が観察されます。

Metaタグ	デスクトップ	モバイル
`includes Referrer-policy`	2.75%	2.52%
`includes CSP`	0.64%	0.59%
`includes not-allowed policy`	0.12%	0.11%

図9.36. metaタグを使用して異なるポリシーを有効にしているホストの割合

他のセキュリティメカニズムは `<meta>` タグを使用して設定できませんが、毎年開発者がこれを試みているのが確認されます。今年はモバイルでmetaタグを使用して設定できないポリシーが0.07%から0.11%に増加しています。これらの値はブラウザに無視されるため、正しいヘッダーが設定されていない場合、ユーザーが脆弱な状態になる可能性があります。継続的な例として、今年は `X-Frame-Options` ポリシーを含む5,564のmetaタグが見つかりました。これは昨年より約600ページ多く、懸念されるトレンドです。

ウェブ暗号化API

ウェブ暗号化API[400]は、基本的な暗号化操作を実行するインターフェースを提供するJavaScript APIです。そのような操作の例は、乱数の生成、ハッシュ化、コンテンツの署名、署名の検証、そしてもちろん暗号化と復号化です。

400. https://www.w3.org/TR/WebCryptoAPI/

機能	デスクトップ	モバイル
CryptoGetRandomValues	34.45%	40.95%
SubtleCryptoDigest	2.65%	2.98%
CryptoAlgorithmSha256	2.36%	2.48%
SubtleCryptoImportKey	1.29%	1.68%
CryptoAlgorithmEcdh	0.97%	1.39%
CryptoAlgorithmSha512	0.17%	0.32%
CryptoAlgorithmSha1	0.21%	0.26%
CryptoAlgorithmAesCbc	0.21%	0.17%
SubtleCryptoSign	0.13%	0.14%
SubtleCryptoEncrypt	0.13%	0.12%
CryptoAlgorithmHmac	0.10%	0.11%

図9.37. ウェブ暗号化APIの機能の使用状況

`CryptoGetRandomValues` 関数はこのAPIで最も広く使用されている機能のままですが、昨年と同様に[401]使用が引き続き低下しています。モバイルでの使用は今年12%以上低下し、41%をわずかに下回っています。他の機能は引き続き上昇しており、2番目に人気の機能 `SubtleCryptoDigest` は1.2%増加して3%弱になっています。

ボット保護サービス

ボットはウェブに長い間存在しており、悪意のあるボットはその大きな部分を占めています。これらの問題のために、異なるベンダーによってウェブサイトをボットから保護するための多くの製品が作成されました。これらの製品の使用は今年も含めて年々増加し続けており、モバイルでの採用率が26.5%から31.1%に急増し、4.5%以上の上昇となっています。

401. *https://almanac.httparchive.org/ja/2024/security#ウェブ暗号化api*

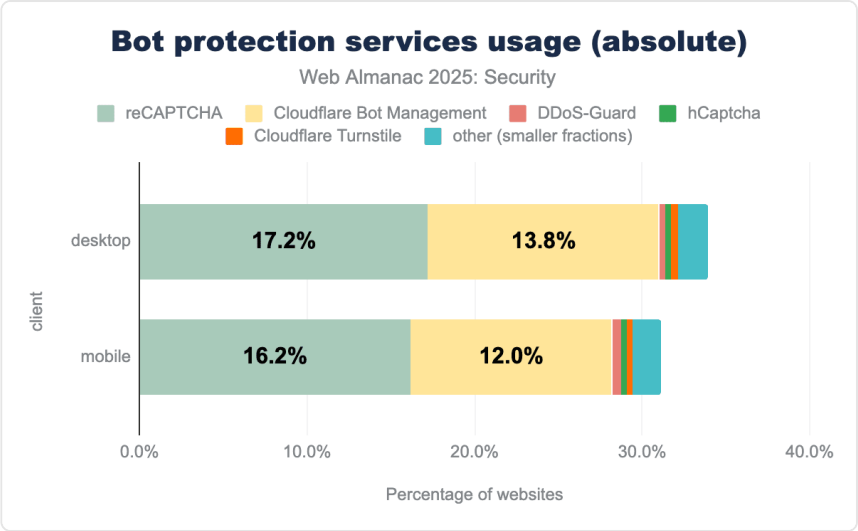


図9.38. 使用中のボット保護サービスの絶対的な分布。

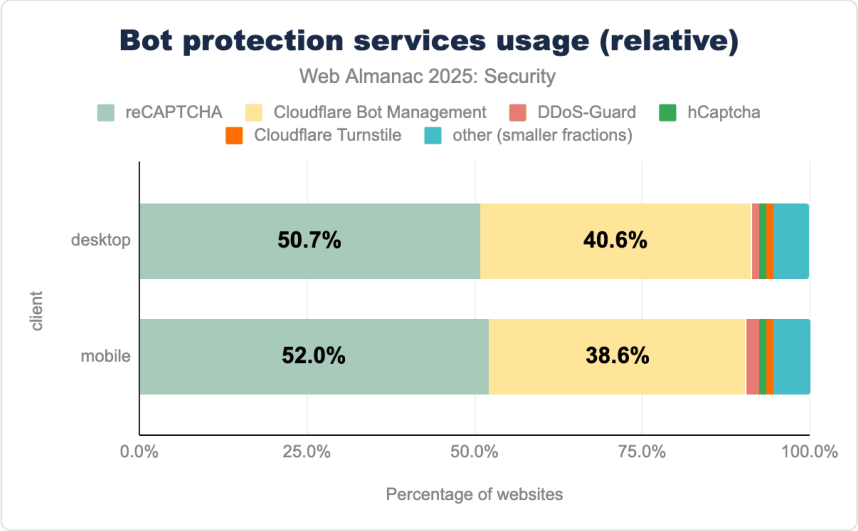


図9.39. 使用中のボット保護サービスの相対的な分布。

reCAPTCHAが依然として最大のボット保護サービスであり続けていますが、Cloudflare Bot Managementはより急速に成長しており、ボット保護サービスを使用するウェブサイトの中でより大きな相対的なシェアを占めています。これらのトレンドが来年も続く場合、Cloudflare Bot ManagementがreCAPTCHAとの差を縮めるのを見るかもしれません。

HTMLサニタイゼーション

`SetHTMLUnsafe` と `ParseHTMLUnsafe` APIは、比較的最近のブラウザへの追加で、開発者はJavaScriptから宣言的なシャドウDOMを使用[402]できます。

開発者が `innerHTML` を使用して宣言的シャドウDOMの定義（例：`<template shadowrootmode="open">...</template>`）を含むカスタムHTMLコンポーネントをページに配置しようとする場合、これは期待通りに機能しません。`SetHTMLUnsafe` または `ParseHTMLUnsafe` を使用することで、開発者は宣言的シャドウDOMがブラウザによって適切にインスタンス化されることを確認できます。

名前が示すように、開発者はこれらの「unsafe」APIに安全な値のみが渡されることを確認する責任があります。そうでない場合、開発者は危険なコンテンツがページに行き着くリスクを負い、XSS攻撃につながる可能性があります。

機能	デスクトップ	モバイル
ParseHTMLUnsafe	19869	17147
SetHTMLUnsafe	443	449

図9.40. HTMLサニタイゼーションAPIを使用するページの数

昨年[403]以降、これらのAPIの使用が大幅に増加しています。モバイルでは、`SetHTMLUnsafe` を使用するページの数が2ページから449ページに増加し、`ParseHTMLUnsafe` を使用するページ数は今年6から17,147に増加しました。後者はクロールされたページの0.06%を占めるに過ぎませんが、興味深い変化であり、翌年も採用が増加し続けることが期待されます。ただし、これらのAPIが近い将来に広く普及することは予想されていません。

セキュリティ機構の採用促進要因

ウェブ開発者がより多くのセキュリティプラクティスを採用することを選ぶ理由はさまざまあります。最も注目すべきものには以下があります：

- **地理的要因**: 地域によって、よりセキュリティ指向の教育や知識があるかもしれません。場合によっては、より厳格なセキュリティ衛生を義務付ける地域の法律がある場合もあります。
- **技術的要因**: 使用する技術スタックはセキュリティ機構の採用に影響を与えるこ

402. *https://developer.chrome.com/blog/new-in-chrome-124#dsd*
403. *https://almanac.httparchive.org/ja/2024/security#htmlサニタイズ*

とがあります。使用する技術によっては、開発者が意識して有効にしなくてもいくつかのセキュリティ機能がデフォルトで有効になります。

- **人気**: 非常に人気のあるウェブサイトは、特定のサイバー攻撃の標的になりやすいため、セキュリティに大きな予算を持っている場合があります。さらに、これらのサイトはより多くのセキュリティ専門家や場合によってはバグバウンティハンターを引き付け、セキュリティ機能の実装と防御の強化を支援します。

ウェブサイトの所在地

ウェブサイトがホストされている場所や開発者が拠点を置いている場所は、特定のセキュリティ機能の採用に大きな影響を与えることがあります。開発者向けのセキュリティ教育は大きな役割を果たしており、開発者は自分が知らない機能や理解していない機能を実装することができません。さらに、地域の法律が特定のセキュリティプラクティスの採用を要求する場合があります。

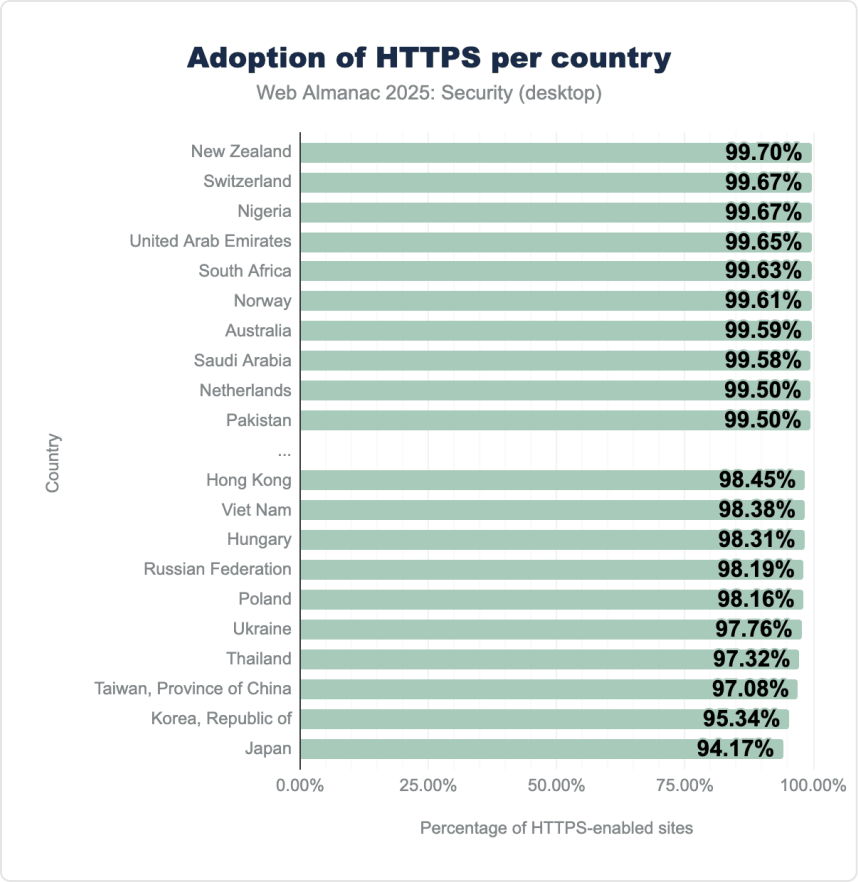


図9.41. 国ごとのHTTPS採用率：採用率の上位10カ国と下位10カ国。

HTTPSの採用は年々増加しており、幸いにも今年もこのトレンドが続いています。上位の国々でさえ採用率が数十分の一パーセントずつ増加し続けているのが見られます。下位の国々ではHTTPS採用率がやや大きく上昇し、日本は今や95%未満の採用率を持つ唯一の国となり、98%未満のHTTPS採用率を持つのはわずか5カ国で、非常に良好な結果です。今後数年で、下位の国々が上位の国々の採用率に徐々に追いつくことが期待されますが、完全な100%の採用は近い将来には難しそうです。

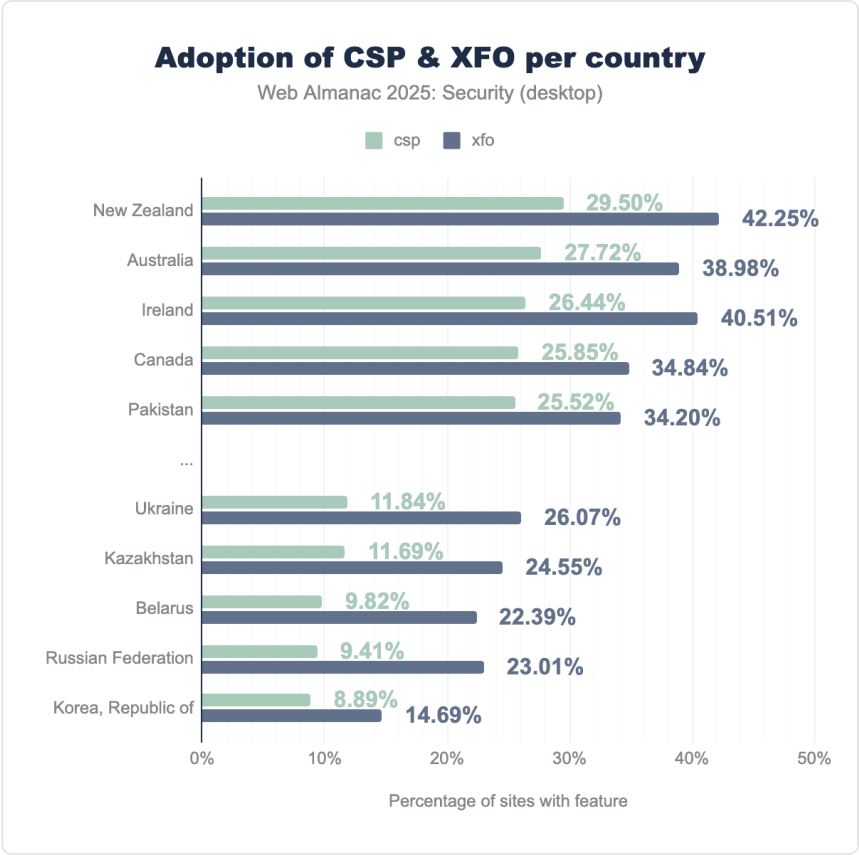


図9.42. 国ごとのCSPとXFOの採用率：CSP採用率の上位5カ国と下位5カ国。

より複雑なセキュリティメカニズムを見ると、より多様な状況が見られます。ほとんどのリーダー国でわずかな採用の増加がある一方で、一部の下位国でこれらのポリシーの採用が減少しました。リーダー国はまだウェブサイトの4分の1強にCSPを設定しています。CSPとXFOの差は大きいままですが、わずかに上昇しており、昨年[404]の15%ではなく最大14%にとどまっています。

テクノロジースタック

ウェブサイトのセキュリティは、使用する技術によって異なる場合があります。フレームワークにはデフォルトでセキュリティ機能が含まれており、大手ベンダーはユーザーをセキュアに保つことが利益になるため、これらの基盤技術を使用することでウェブサイトのセキュ

404. *https://almanac.httparchive.org/ja/2024/security#ウェブサイトの場所*

リティが向上する可能性があります。

テクノロジー	セキュリティ機能
LiveJournal	`Content-Security-Policy` (99.99%), `Permissions-Policy` (99.99%), `Referrer-Policy` (99.99%)
Weblium	`X-Content-Type-Options` (97.31%), `X-XSS-Protection` (97.31%)
GoDaddy Website Builder	`Strict-Transport-Security` (95.97%), `Content-Security-Policy` (95.97%)
Sitevision CMS	`X-Frame-Options` (81.54%)
Microsoft Sharepoint	`X-Content-Type-Options` (57.44%)
Liferay	`X-Content-Type-Options` (52.65%)

図9.43. 選択されたCMSシステムで使用されているセキュリティ機能

多くのブログウェブサイトとウェブサイトビルダーが、HSTS、CSP、`X-Content-Type-Options` が最も人気のあるものの中に入る、いくつかの重要なセキュリティメカニズムをほぼすべてのシステムで設定していることが分かります。

ウェブサイトの人気

大規模なユーザーベースを持つ人気サイトは、ユーザーとその信頼を失わないように、最善の方法でユーザーを保護する十分な理由を持っていることが多いです。しばしば機密性の高いデータを保護しながら、より標的を絞った攻撃の対象になる可能性が高い中で、ウェブサイトのセキュリティへの多大な投資が必要ですが、そのトレードオフとして、より一般的にセキュアなウェブサイトにつながります。

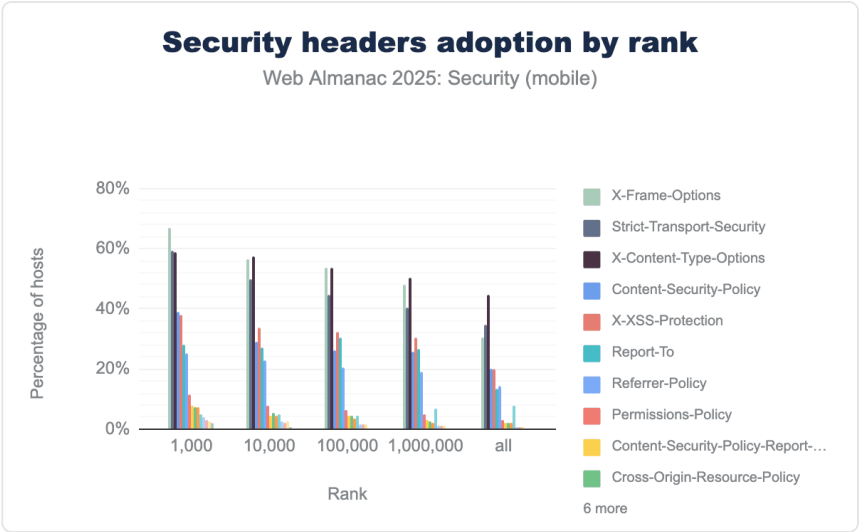


図9.44. CrUXによるウェブサイトランク別のセキュリティヘッダー採用率。

`X-Frame-Options`、`Strict-Transport-Security`、`X-Content-Type-Options`、`Content-Security-Policy`などの最も人気のあるセキュリティヘッダーは、常にモバイルでより人気のあるウェブサイトでより高い採用率を示しています。最も広く採用されているヘッダーである`X-Frame-Options`は上位1,000サイトの67%で使用されていますが、ブラウザが訪問するすべてのサイトの30%でしか使用されていません。より人気のあるサイトとそうでないサイトの採用率の差は、昨年[405]から事実上同じままであることが分かります。

ウェブサイトカテゴリー

業界によっては、ウェブサイトをよりセキュアに保つことに対してより多くの重要性が帰属される場合があります。ウェブサイトが設定するセキュリティヘッダーの平均数に基づいて、業界ごとのウェブサイトのセキュリティへの取り組みを推定しようとします。セキュリティヘッダーの数は必ずしもウェブサイトがより良くセキュアされているかどうかを示すわけではありませんが（セキュリティメカニズムは誤設定される可能性があります）、セキュリティ機能を実装するための取り組みの良い指標を提供します。

405. *https://almanac.httparchive.org/ja/2024/security#ウェブサイトの人気*

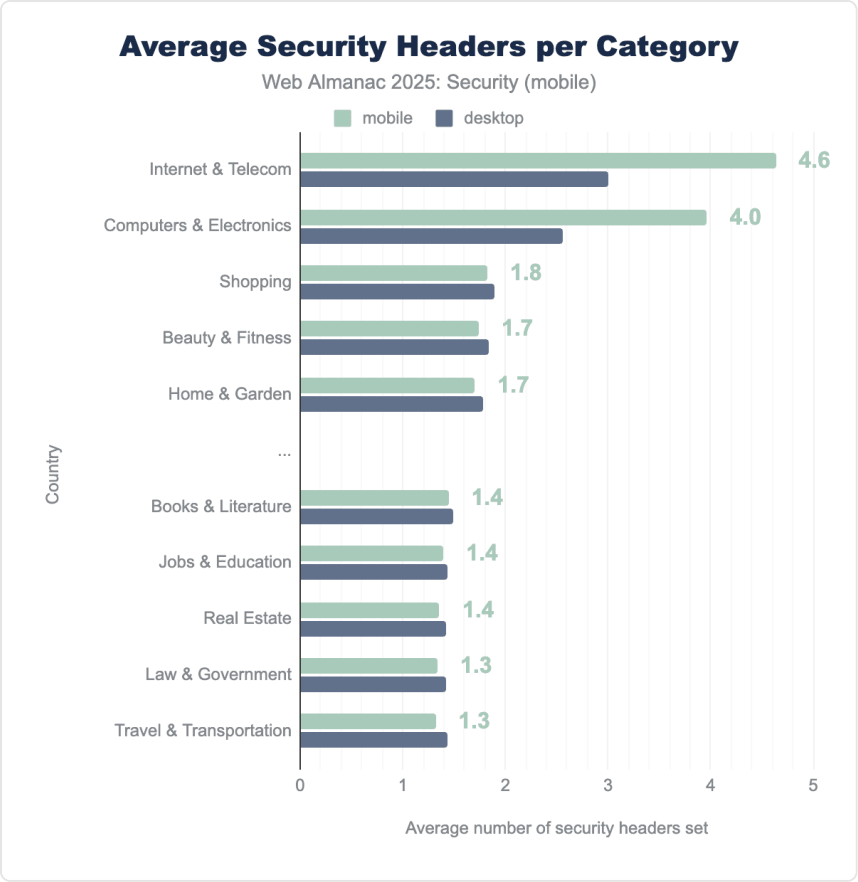


図9.45. ウェブサイトカテゴリー別のセキュリティヘッダーの平均数：上位5カテゴリーと下位5カテゴリー。

昨年からセキュリティヘッダーの平均数に大きな変化が見られます。 *インターネット＆テレコム* と *コンピューター＆エレクトロニクス* カテゴリーのサイトはより多くのセキュリティヘッダーを使用しています。特にモバイルクライアントではこれらのカテゴリーとの差が明確に見えます。これら2つのカテゴリーが良い意味での例外的なケースであるように見える一方で、他の業界のセキュリティヘッダーの平均数はほぼ同じままで、サイトあたり平均約0.1ヘッダーのわずかな変化があるだけです。

セキュリティヘッダーの観点でリードする2つの業界は、インターネットとコンピューターセキュリティの分野に関連した業界です。これらの業界ではセキュリティの関連性が高いため、これらのサイトの開発者は潜在的なリスクをより認識しており、特定のセキュリティメカニズムをより積極的に使用することが考えられます。

ウェブ上の不正行為

暗号通貨はこれまで以上に人気があります。暗号通貨マイニングは何年もの間大きなビジネスであり、攻撃者が被害者のウェブサイトにマルウェアの一形態として暗号マイナーをインストールすることが知られています。過去数年で、ウェブ上での暗号マイナーの使用は着実に減少しています。

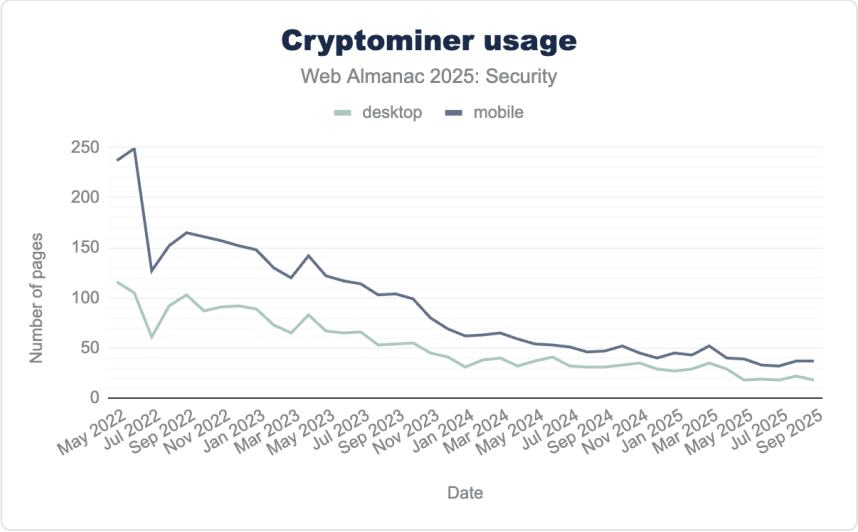


図9.46. 経時的な使用中の暗号マイナー数：2022年5月から2025年9月まで。

暗号マイナーを持つページの数はモバイルでわずか37ページに減少しており、昨年[406]から42%の減少です。これはまた、わずか3年前の2022年9月から83%の減少でもあります。

406. https://almanac.httparchive.org/ja/2024/security#fig-47

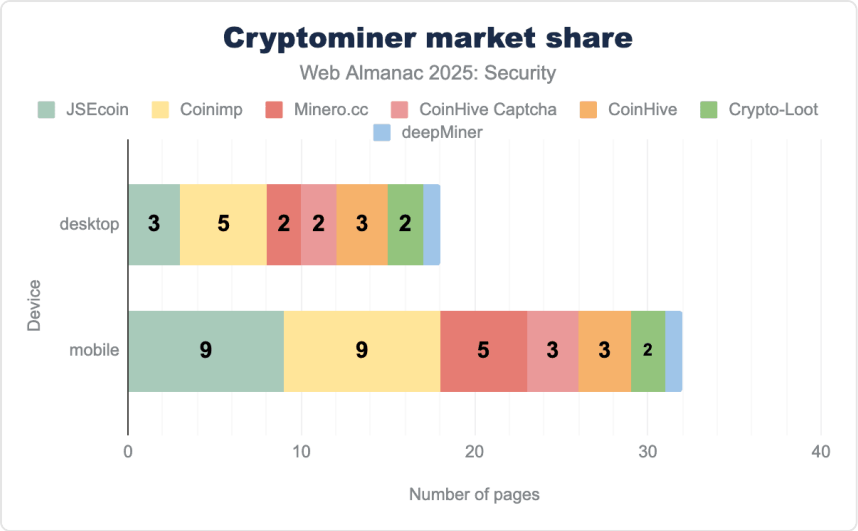


図9.47. 暗号マイナーの市場シェア。

ウェブ上の暗号マイナーの絶対数は非常に少ないものの、異なる暗号マイナーが代表するシェアを確認します。昨年[407]と比較して、Coinimpを持つページの数がJSEcoinを持つ9ページと一致するまで減少しています。興味深い点は、デスクトップとモバイルページの暗号マイナーの数の違いで、モバイルのページ数はデスクトップのほぼ2倍になっています。

セキュリティの誤設定と見落とし

ウェブとブラウザで利用可能なセキュリティメカニズムは数多くありますが、これらのメカニズムを正しく期待通りに設定することが不可欠です。誤設定されたセキュリティメカニズムは、ユーザーが保護されていると思っている開発者に偽りのセキュリティ感を生み出します。このセクションでは、ウェブサイトのセキュリティを損なう可能性のあるいくつかの誤設定の発生を強調します。

`<meta>`でサポートされていないポリシー

セキュリティポリシーを設定する際、開発者はポリシーをどのように定義すべきかを理解することが重要です。一部のポリシーはヘッダーとHTML `<meta>` タグの両方で定義できます。ただし、多くのポリシーは `<meta>` を通じて定義できません。開発者は時々、これらのポリシーを `<meta>` タグを通じて設定しようとする間違いを犯します。残念ながらブラ

407. https://almanac.httparchive.org/ja/2024/security#fig-48

ウザはこれらのポリシーを無視し、非アクティブなセキュリティポリシーになります。

ポリシー	デスクトップ	モバイル
X-Frame-Options	4,584	5,564
X-Content-Type-Options	2,440	2,854
Permissions-Policy	1,983	2,236
X-XSS-Protection	1,691	1,702
Referrer-Policy	1,630	1,644

図9.48. `<meta>`を通じて誤って設定されたセキュリティポリシーの上位5件

モバイルサイトの約0.11%が、ブラウザが `<meta>` タグを通じて明示的にサポートしないセキュリティヘッダーを設定しようとしていることが分かります。最もよく試みられるポリシーは `X-Frame-Options`、`X-Content-Type-Options`、`Permissions-Policy` です。2024年と比較して、`<meta>` でこれらのポリシーを設定するモバイルサイトの数が絶対数で5,000ページ以上増加しており、これらの誤設定が開発者によってまだ積極的に設定されていることを示しています。

`<meta>`でサポートされていないCSPディレクティブ

CSPは `<meta>` タグを通じて設定できますが、この動作はモバイルページの0.59%で観察されており、特定のCSPディレクティブは `<meta>` タグの実装では明示的に禁止されており、ブラウザに無視されます。これらのディレクティブは `Content-Security-Policy` レスポンスヘッダーを使用することでのみ設定できます。

ディレクティブ	デスクトップ	モバイル
frame-ancestors	2.37%	2.11%
sandbox	0.004%	0.003%

図9.49. `<meta>`で定義されたCSPポリシーのうち禁止されたディレクティブを含む割合

`<meta>` タグを通じてCSPポリシーを設定するモバイルページの2%以上が `frame-ancestors` ディレクティブを含んでおり、0.003%が `sandbox` ディレクティブを含んでいることが分かります。後者はクロールされたデータセット全体でわずか3ページに相当します。前回版と比較して、`frame-ancestors` の誤設定は600ページ多く現れており、0.8%以上上昇しています。これはこの種の誤設定に対するゆっくりだが否定的な進展を表していま

す。

COOP/COEP/CORPの混同

クロスオリジンポリシーのCOEP、CORP、COOPは類似した命名を持ち、目的が関連しているため、開発者によって混同されることがあります。これらのヘッダーに誤った値を割り当てると、ブラウザはヘッダーがまったく提供されていないかのようにデフォルトのポリシーを適用し、開発者が望む追加の防御が無効になります。

無効なCOEP値	デスクトップ	モバイル
`same-origin`	4.43%	4.02%
`cross-origin`	0.55%	0.46%
*	0.13%	0.11%
`(unsafe-none|require-corp); report-to='default'`	0.09%	0.11%
`: require-corp`	0.09%	0.10%

図9.50. 無効なCOEPヘッダー値の普及率

合計で、観察されたCOEPヘッダーの5.6%がモバイルで無効な値を含んでいます。これらのヘッダーの4%以上がCORPまたはCOOPヘッダーでのみ有効な値である `same-origin` を含んでいます。別の0.5%近くはCORPヘッダー用の値である `cross-origin` を含んでいます。残念ながら、これらの誤設定は昨年[408]からも増加し、COEPヘッダーの `same-origin` 値で約1%上昇しました。

これらの誤設定に加えて、ブラウザが解析できないため、ヘッダーのデフォルト値に戻ることになる構文エラーを含むいくつかの値も観察しました。これらの構文エラーは少数のケースを表しています。

Timing-Allow-Originワイルドカード

`Timing-Allow-Origin`（TAO）レスポンスヘッダーはサーバーがResource Timing API[409]の機能を通じて取得した属性の値にアクセスできるオリジンのリストを指定できるようにします。このヘッダーにリストされたオリジンは、TCPコネクションの開始時間、リクエストの開始、レスポンスの開始など、サーバーへの接続に関する詳細なタイムスタンプにアクセスできます。このアクセスをオリジンに許可する際は、リストされたオリジンがタイミング攻

408. *https://almanac.httparchive.org/ja/2024/security#coepcorpcoopの混同*
409. *https://developer.mozilla.org/docs/Web/API/Performance_API/Resource_timing*

撃や他のクロスサイト攻撃を実行する可能性が生まれるため、慎重に行う必要があります。

CORSが設定されている場合、上記のようなこれらの種類のタイミング値の多くはクロスオリジンリークを防ぐために0として返されます。`Timing-Allow-Origin` ヘッダーにオリジンをリストすることで、これらの値は元の値を保持し、ゼロアウトされなくなります。

図9.51. `Timing-Allow-Origin` ヘッダーがワイルドカード（`*`）値に設定されている割合。

オリジンのリストを返すことに加えて、開発者は `Timing-Allowed-Origin` ヘッダーの値をワイルドカード文字 `*` に設定して、どのオリジンでもタイミング情報にアクセスできることを示すことができます。TAOヘッダーの84.6%がワイルドカード `*` 値を含んでおり、昨年[410]から2%上昇していることが分かります。これは多くの開発者がきめ細かなタイミング情報を任意のオリジンに普遍的に公開することに問題がないことを示しています。

サーバー情報ヘッダーの抑制の欠如

ウェブサイトがサーバーとその特定バージョンなど、インフラについての過剰な情報を公開した場合、自動脆弱性スキャナーの標的になるリスクが高くなる可能性があります。この情報を非表示にすることはセキュリティ・バイ・オブスキュリティの一形態であり、システムのコアな脆弱性には対処しないため一般的に眉をひそめられる戦略ですが、特定の攻撃のレーダー下に留まるのに役立つ可能性があるため分析を含めます。

このような情報を報告するために一般的に使用されるヘッダー、すなわち：`Server`、`X-Server`、`X-Backend-Server`、`X-Powered-By`、`X-Aspnet-Version` の使用を追跡します。

410. *https://almanac.httparchive.org/ja/2024/security#timing-allow-originワイルドカード*

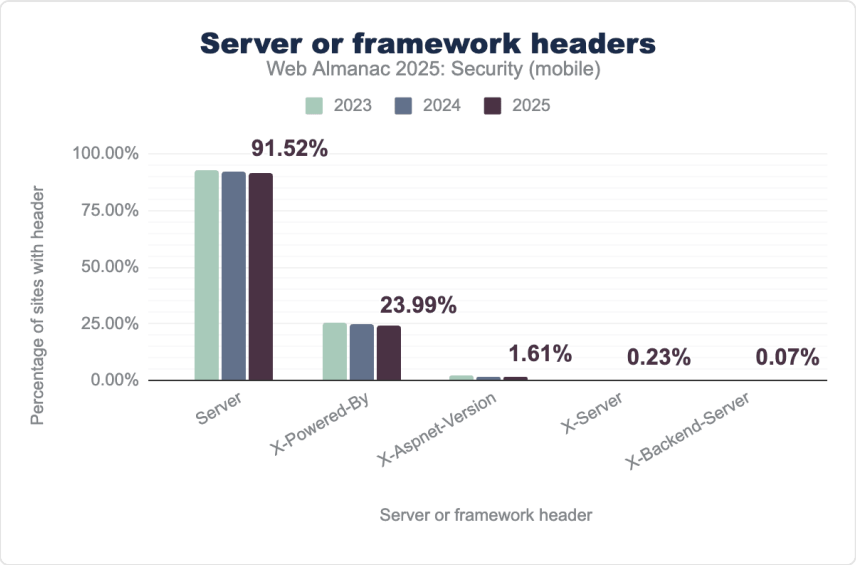


図9.52. サーバーに関する情報を伝えるために使用されるヘッダーの普及率。

過去の年と同様に、`Server` ヘッダーは `X-Powered-By` を大差で最も広く使用されているヘッダーであり続けています。これらのヘッダーはウェブ上のホストの91.5%と23.9%に表示されます。各ヘッダーについて、表示されるホストの数がわずかに減少しているのが見られます。多くのウェブ技術がこれらのヘッダーの一部を自動的に設定し、開発者はセキュリティ上の利点が小さいため、これらのヘッダーを削除することにあまり興味がないかもしれないため、これらの値が時間とともに大きく変化することは期待されません。

ヘッダー値	デスクトップ	モバイル
PHP/7.4.33	9.54%	9.98%
PHP/7.3.33	3.61%	4.29%
PHP/5.3.3	2.10%	2.20%
PHP/5.6.40	2.07%	2.12%
PHP/8.0.30	1.55%	1.70%
PHP/7.2.34	1.34%	1.41%
PHP/8.2.28	1.15%	1.31%
PHP/8.3.13	1.08%	1.11%
PHP/8.1.32	1.05%	1.09%
PHP/8.1.27	0.92%	1.06%

図9.53. 特定のフレームワークバージョンを含む最も普及している`X-Powered-By`ヘッダー値

`Server`と`X-Powered-By`値の中で最も一般的な値を確認すると、PHPが特に`X-Powered-By`ヘッダーでサーバー上で動作している正確なバージョンを公開する傾向があることが分かります。デスクトップとモバイルの両方で、`X-Powered-By`ヘッダーの27%以上がバージョン情報を含んでいることが分かります。ヘッダーはデータで観察されるプラットフォームによって自動的に返される可能性があります。興味深いことに、古いPHPバージョン7.xと低いバージョンのわずかな減少と、新しいPHPバージョン8.xのわずかな増加が見られ、少なくとも一部の開発者がサーバーをアップデートしていることの兆しです。

`Server-Timing`ヘッダーの抑制の欠如

`Server-Timing` HTTPヘッダーはサーバーパフォーマンスメトリクスを通信するために使用できるW3C Editor's Draft[411]で定義されたレスポンスヘッダーです。開発者はゼロ以上のプロパティを含むメトリクスを送信できます。指定されたプロパティの一つは`dur`プロパティで、サーバー上の特定のアクションの持続時間を含むミリ秒精度のタイミングを通信するために使用できます。

411. *https://w3c.github.io/server-timing/*

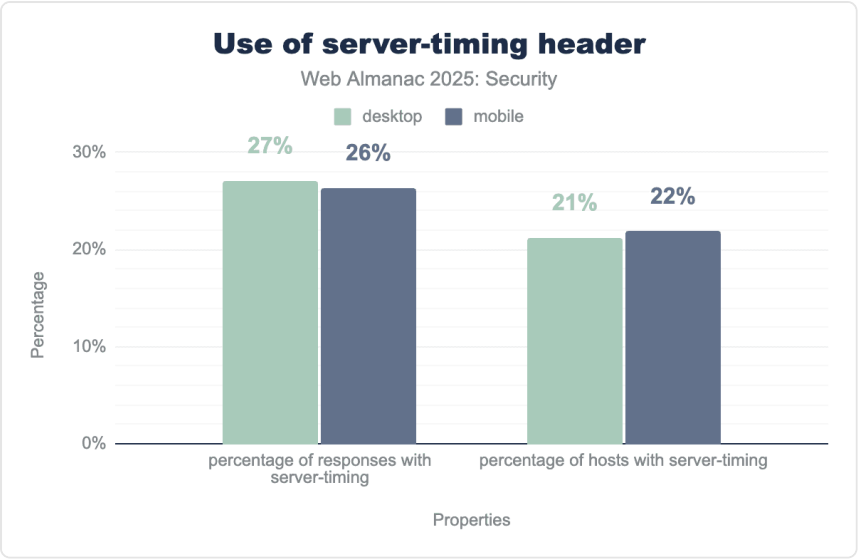


図9.54. `Server-Timing` *ヘッダーの使用状況。*

`server-timing` ヘッダーを返すホストの割合は、クローラーが訪問したホストの5分の1以上で15%以上増加しました。これは昨年[412]からの非常に急激な増加です。

412. *https://almanac.httparchive.org/ja/2024/security#server-timingヘッダーの抑制の欠如*

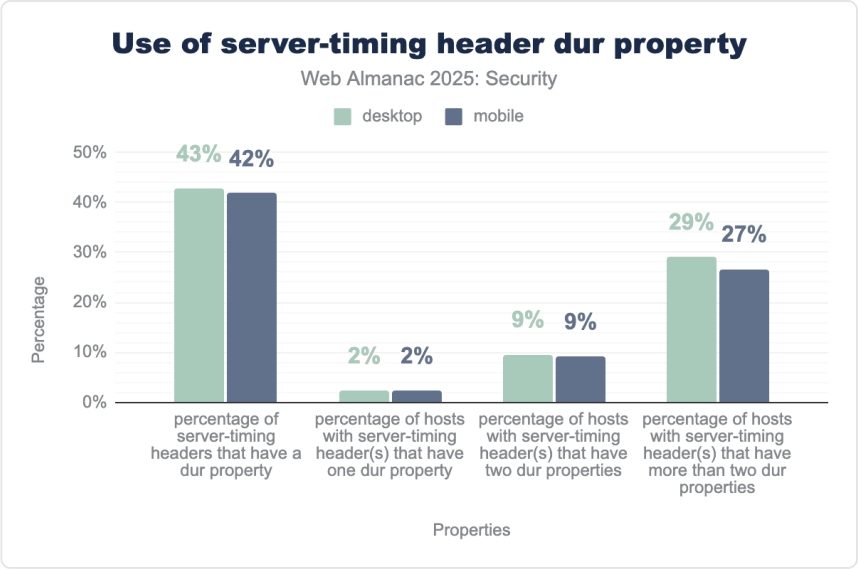


図9.55. `Server-Timing` ヘッダーの `dur` プロパティの相対的な使用状況。

ウェブ上の `server-timing` ヘッダーのうち、42%が少なくとも1つの `dur` プロパティを持っていることが分かります。これは昨年[413]と比較した場合の相対的な減少ですが、年間のヘッダー使用量の急激な増加を考えると、絶対数は増加しています。これはまた、より多くのヘッダーが `dur` プロパティを含まず、開発者が特定のメトリクスの説明を設定できる `desc` プロパティの使用などを通じて、ヘッダーを他の目的で使用していることを意味します。

`server-timing` ヘッダーに含まれる情報は機密性がある可能性があるため、値へのアクセスは同一オリジンと `Timing-Allow-Origin` ヘッダーにリストされたオリジンに制限されています。上述のように、多くのウェブサイトはワイルドカード文字で `Timing-Allow-Origin` を設定しており、潜在的に機密性の高い情報にすべてのオリジンがアクセスできるようにしています。クロスオリジンアクセスがなくても、ブラウザのコンテキスト外でサーバーが機密性の高いタイミング情報を公開している場合、タイミング攻撃をサーバーに直接実行することが可能です。

Well-known URI

Well-known URIはサイト全体のメタデータとサービスのための特定の場所を指定するための

413. *https://almanac.httparchive.org/ja/2024/security#fig-53*

標準化されたメカニズムを提供します。RFC 8615[414]で定義されているwell-known URIは、プレフィックス `/.well-known/` で始まるパスコンポーネントによって識別されます。これにより、クライアントはサイトのURLスキームに関する事前知識がなくても特定のリソースを発見できます。

`security.txt`

よく知られている `security.txt` ファイルはウェブサイトが脆弱性報告情報を通信するために使用する標準化されたファイル形式です。ホワイトハットハッカーやペネトレーションテスターはこのファイルを使用して、責任ある開示のための連絡先詳細、PGPキー、ポリシー、その他の情報を見つけることができます。

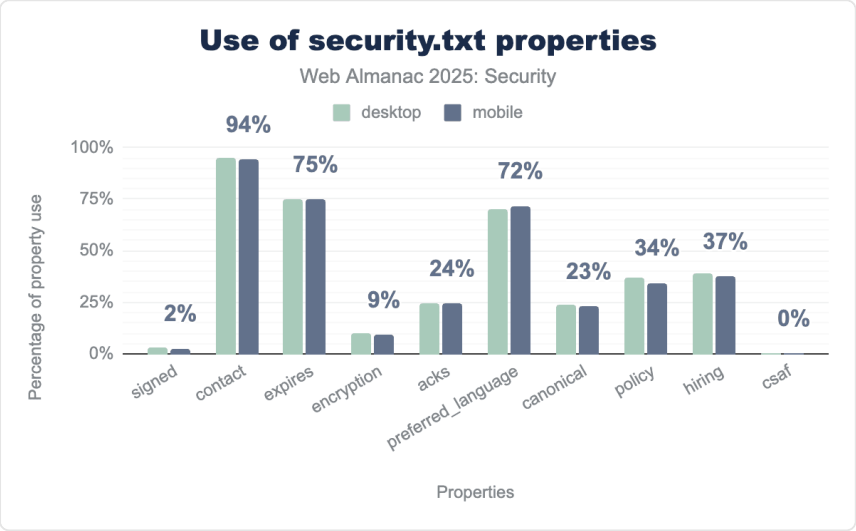


図9.56. `security.txt` プロパティの使用状況。

採用率はデスクトップで1.82%、モバイルサイトで1.72%に増加し、どちらも2024年の1%[415]から上昇しており、標準化されたセキュリティ開示プラクティスへの認識が高まっていることを示しています。

`security.txt` を実装しているサイトの中では、連絡先情報がデスクトップで95%、モバイルで94%とほぼ普遍的に含まれており、2024年の92%と89%からそれぞれ上昇しています。興味深いことに、75%が有効期限を定義しており、2024年のデスクトップで51%、モバイルで48%から大幅に増加しています。優先言語は実装の70〜72%で指定されており、脆弱

414. https://www.rfc-editor.org/rfc/pdfrfc/rfc8615.txt.pdf
415. https://almanac.httparchive.org/ja/2024/security#securitytxt

性報告手順を定義するポリシーはデスクトップで37%、モバイルファイルで34%にのみ現れており、2024年の39%から低下しています。ただし、ポリシーを定義する `security.txt` ファイルの絶対数は3分の2増加しました。

この分析は、`security.txt` ファイルの少なくとも25%が完全に有効ではないことを示しています。これはRFC 9116[416]で規定されているように、連絡先情報とともに有効期限を含めることが必要だからです。

`change-password`

`change-password` well-known URIは2021年のW3C仕様草案で、それ以来更新されていません。このURIの目的は、ユーザーと外部リソースが特定のサイトのパスワードを変更できる正しい場所をすばやく見つけるためのものです。

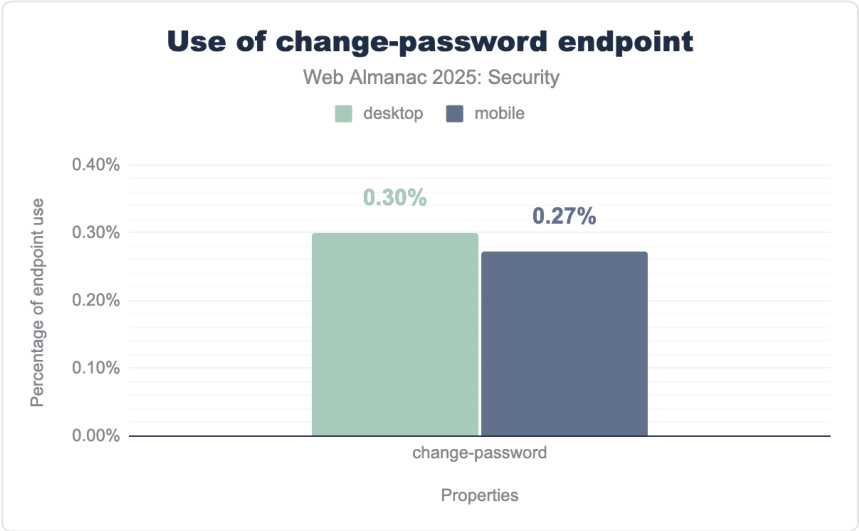


図9.57. change-password .well-knownエンドポイントの使用状況。

デスクトップサイトは0.27%から0.30%にわずかに上昇し、モバイルエンドポイントでは0.27%のまま変化がないことが分かります。この緩やかな採用は、すべてのウェブサイトが認証メカニズムを必要とするわけではないことを考えると、予想外ではありません。

416. https://www.rfc-editor.org/rfc/rfc9116.txt

ステータスコードの信頼性検出

ウェブサイトのHTTPレスポンスステータスコードの信頼性を確認するための仕様[417]草案も2021年から変更されていません。この特定のwell-knownエンドポイントの目的は、それが決して存在してはならず、したがってレスポンスステータスコードが決してokステータス[418]であってはならないというものです。サイトがokステータスで応答するリダイレクトが発生した場合、これは不正な動作とみなされます。

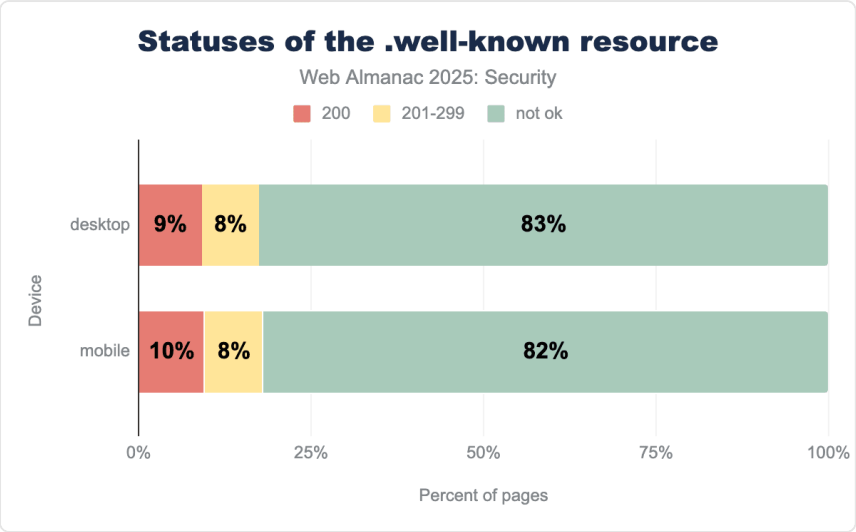


図9.58. ステータスコードの信頼性を評価するための `.well-known` エンドポイントに返されるステータスの分布。

2022年版[419]と2024年版[420]のWeb Almanacと同様の結果が見られます。ただし、誤ったokステータスレスポンスの数はわずかに増加しており、2024年のデスクトップとモバイルページの両方で84%からそれぞれ83%と84%になっています。ウェブ開発者は、これらのステータスコードが意味を失わないようにするために、アプリケーションが受信リクエストに正しいステータスコードで応答し続けるようにする必要があります。

`robots.txt` 内の機密エンドポイント

最後に、よく知られている `robots.txt` ファイルで機密エンドポイントがクローラーによる訪問を禁止されているかどうかを確認します。このファイルの禁止されたエンドポイント

417. *https://w3c.github.io/webappsec-change-password-url/response-code-reliability.html*
418. *https://fetch.spec.whatwg.org/#ok-status*
419. *https://almanac.httparchive.org/ja/2022/security#change-password*
420. *https://almanac.httparchive.org/ja/2024/security#change-password*

を確認することで、攻撃者は標的にするページを見つける可能性があります。今年は、デスクトップサイトの81%とモバイルサイトの80%が `robots.txt` ファイルをホストしており、昨年版のWeb Almanac[421]と非常に似ています。

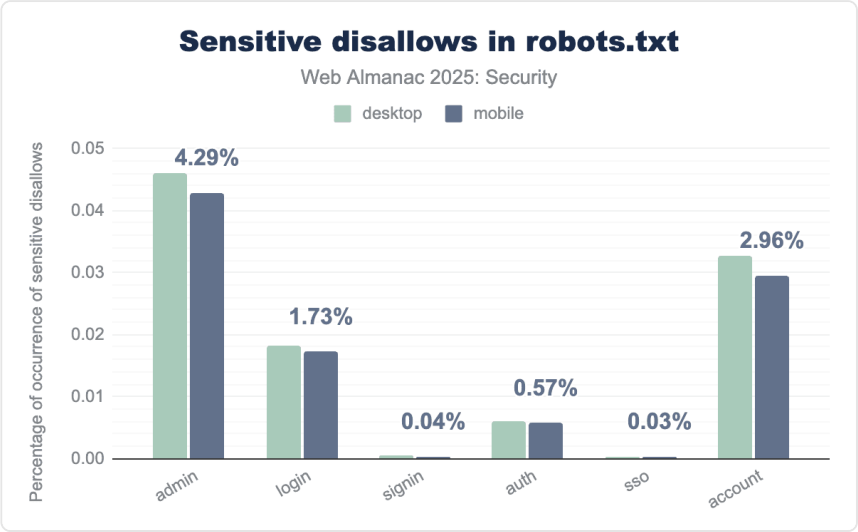


図9.59. 指定されたエンドポイントを `robots.txt` に含むサイトの割合。

これらのファイルの内容も非常に似ています。最大の増加はデスクトップサイトで `auth` エンドポイントの0.2%であり、最大の減少はデスクトップサイトで `login` エンドポイントの0.2%でした。

結論

このセキュリティチャプターはウェブセキュリティポリシーの採用において前向きなトレンドを示しています。HTTPSは全体的に100%近くの採用率に達しており、国別のメトリクスは全ての国がHTTPSの普遍的な使用という目標に向けて進んでいることを示しています。`Content-Security-Policy` は18%以上の使用増加、`Permissions-Policy` は昨年より50%多く使用されるなど、最新の攻撃からユーザーをより良く保護することを目的とした多くの最新セキュリティポリシーの採用が増加しているのが見られました。また、ドキュメントポリシーのような新しいポリシーが実際の環境に現れているのも見られ、開発者が新しいセキュリティ機能の採用に積極的に取り組んでいることを示しています。

これらの前向きなトレンドにもかかわらず、開発者はこれらのセキュリティメカニズムを活

421. *https://almanac.httparchive.org/ja/2024/security#robotstxtの機密エンドポイント*

用する際に警戒を続けなければなりません。利用可能な多くのセキュリティメカニズムの複雑さの増大により、ウェブ上での誤設定の数が増加しているのが見られました。ページの0.1%がブラウザでサポートされていないにもかかわらず `<meta>` HTMLタグでセキュリティポリシーを設定していることが分かりました。もう一つの問題は関連する保護の間の混同です：COEPヘッダーの値の5%は、関連するCORPまたはCOOPヘッダーでのみ有効な無効値です。また、開発者の疲労のような形も観察されており、デプロイをより管理しやすくするか潜在的な問題を防ぐために、保護の最も緩やかな値が設定されています。例えば `Timing-Allow-Origin` ヘッダーのワイルドカード値はこれらのヘッダーの84%以上に現れています。幸い、開発者が問題を認識すれば、これらの問題を簡単に軽減できます。

将来の新しい攻撃は必然的に世界中のユーザーを安全に保つためのさらに多くの保護メカニズムの設計を推進するでしょう。政策立案者は開発者の混乱を避けるためにこれらの新しいメカニズムの複雑さを減らすことに焦点を当てる必要がありますが、新しいセキュリティ機能の採用に時間がかかる一方で、比較的新しいポリシーが採用され、時間とともに多くの採用を得ており、すべての人にとってより安全なウェブを生み出しています。

著者

Vik Vanderlinden

@vikvanderlinden vikvanderlinden vikvanderlinden https://vikvanderlinden.be/

Vik VanderlindenはKU LeuvenのDistriNet研究グループ[422]のコンピュータサイエンス博士課程の学生です。ウェブとネットワークセキュリティの研究に取り組んでおり、特にウェブアプリケーションとプロトコルにおけるタイミングリークに注力しています。

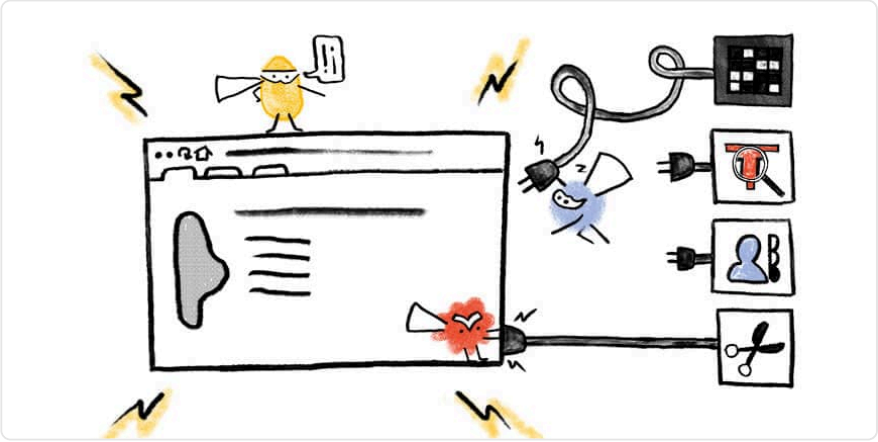
Gertjan Franken

@GJFR_ GJFR gertjan-franken https://gjfr.dev

Gertjan Frankenは、KU LeuvenのDistriNet研究グループ[423]のポスドク研究員です。ウェブセキュリティとプライバシーの様々な側面を研究しており、特にブラウザセキュリティポリシーの自動分析に注力しています。この研究の一環として、バグのライフサイクルを特定するオープンソースツールBugHog[424]を管理しています。

422. *https://distrinet.cs.kuleuven.be/*
423. *https://distrinet.cs.kuleuven.be/*
424. *https://github.com/DistriNet/BugHog*

部 II 章 10
ケイパビリティ



***Alba Silvente Fuentes** によって書かれた。*
Maxim Salnikov と Thomas Steiner によってレビュ。
Estela Franco と Thomas Steiner による分析。
Barry Pollard 編集。
Sakae Kotaro によって翻訳された。

はじめに

今日のウェブブラウザは、これまでにないほど豊かなウェブ体験を提供しています。ブラウザ自体の基本的な機能に留まらず、低レベルの機能や実行中のオペレーティングシステムも活用しています。

こうしたケイパビリティは、Clipboard[425]、File System[426]、Service Worker[427] などの実績あるAPIから、ウェブアプリの開発を変革する可能性を持つ実験段階の新しいAPIまで、ウェブプラットフォームAPIを通じて提供されています。

AIの時代において、ブラウザは取り残されるわけにはいきません。AIの民主化のために、誰もが利用できる持続可能でアクセシブルなAI APIを提案していく必要があります。そこで今年のケイパビリティチャプターでは、ChromeとEdge固有のこれらの新しいAPIの初期利用

425. https://developer.mozilla.org/docs/Web/API/Clipboard_API
426. https://developer.mozilla.org/docs/Web/API/File_System_API
427. https://developer.mozilla.org/docs/Web/API/Service_Worker_API

状況について取り上げます。

方法論

本チャプターは例年同様、HTTP Archiveの数百万ページに及ぶ公開データセットを使用しています。サイトによってはリクエスト元のデバイスによって異なるコンテンツを提供するため、デスクトップとモバイルの両方でアーカイブされています。

HTTP Archiveはどのようにケイパビリティ APIを検出しているか？

HTTP Archiveのクローラーは、`/navigator\.share\s*\(/g` のような正規表現を使って、各ページでどのAPIが（潜在的に）使用されているかを判定するためにソースコードを解析します。

この検出方法にはいくつかの問題があります。例えばミニファイによって `navigator` が `n` に短縮されている場合など、検出できないAPIが存在するため過少報告になる場合があります。一方、実際にAPIが使用されているかどうかをコードを実行して確認するわけではないため、過剰報告になる場合もあります。

こうした制限はあるものの、本チャプターの他のエディションと同様、現在ウェブ上でどのようなケイパビリティが使われているかについて、十分な概観を得ることができます。

サポートされるケイパビリティに対して合計86の正規表現が存在します。使用されている全ての式は、Fugu API data[428]プロジェクトをベースにしたソースファイル[429]で確認できます。

Project Fugu

データを詳しく見ていく前に、ウェブとモバイル/デスクトップアプリケーション間の機能同等性の実現を目指すクロスカンパニーイニシアチブであるProject Fuguへの感謝を表明したいと思います。

このイニシアチブのおかげで、プラットフォーム固有のケイパビリティをウェブに公開することで、これまでアプリケーションのみで利用可能だった多くの機能を活用できるようになっています。

このコンテキストで公開されているAPIについては、Chromeデベロッパーブログでケイパビリティプロジェクト[430]の詳細をご覧ください。

428. *https://github.com/tomayac/fugu-api-data*
429. *https://github.com/HTTPArchive/custom-metrics/blob/main/dist/fugu-apis.js*
430. *https://developer.chrome.com/docs/capabilities*

最もよく使われる機能トップ7

以下のセクションでは、2025年のデータセットで観測されたウェブプラットフォームケイパビリティの中で最も広く使用されている7つを取り上げます。これらの機能は、長年使われてきたAPIと、広く実用的に採用されるまでに成熟した比較的新しいAPIが混在しています。モバイルとデスクトップの両方のページで普及していることは、今日のウェブプラットフォームが最も多く依存している領域を示しており、現代のウェブアプリケーションの基礎的な構成要素となっているケイパビリティを理解するための有用な基準を提供します。

Compression Streams API

Compression Streams API[431]は、ウェブアプリがGZIPやDeflate（最近ではBrotliも）などの広くサポートされたフォーマットを使用して、ブラウザ上でデータを直接圧縮・解凍できるようにします。これにより、サーバーサイドの処理に依存せず、大量データの効率的な転送と保存が可能になります。

データは `CompressionStream` と `DecompressionStream` オブジェクトを通じて処理され、ウェブのStreaming API[432]（`ReadableStream`、`WritableStream`）と統合されます。

```
const text = "Hello Web Almanac 2025!";
const stream = new Blob([text]).stream();
const compressed = stream.pipeThrough(new
CompressionStream("gzip"));
const decompressed = compressed.pipeThrough(new
DecompressionStream("gzip"));
const result = await new Response(decompressed).text();

console.log(result); // "Hello Web Almanac 2025!"
```

2023年5月以降、この機能は最新のデバイスとブラウザバージョンで動作します。Chromiumベースのブラウザ、Safari、Firefoxで利用可能ですが、古いデバイスや他のブラウザでは動作しない場合があります。

431. *https://developer.mozilla.org/docs/Web/API/Compression_Streams_API*
432. *https://web.dev/streams*

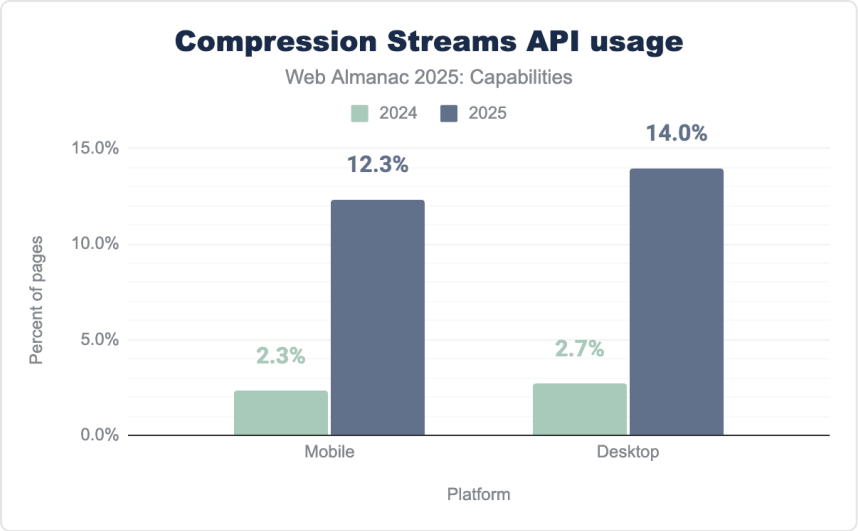


図10.1. Compression Streams APIの使用状況（2024年〜2025年）。

Compression Streams APIの採用は2024年から2025年にかけて急増し、2025年に最も広く使用されるAPIとなり、3年間首位だったClipboard APIを抜きました。

モバイルでは使用率が2.3%から12.3%に、デスクトップでは2.7%から14.0%に急増しました。この急激な上昇は、過去2年間でAPIが主要なエンジン全てで広くサポートされる[433]ようになり、技術的な障壁が取り除かれ、開発者がJavaScriptのポリフィルを削除してネイティブのgzip/deflate圧縮を利用できるようになったことと一致しています。

このAPIはストリーミング効率が重要なデータ量の多いアプリケーションに特に有用であり、それが強い採用曲線を説明しています。

Clipboard API

Clipboard API[434]は、システムクリップボードへの非同期の読み書きアクセスを提供します。プレーンテキスト、HTML、画像などのフォーマットをサポートしていますが、ブラウザによってサポート状況が異なる場合があります。

セキュリティ上の制限[435]により、クリップボード操作はクリックなどのユーザーアクションをトリガーとする必要があります。

433. *https://web.dev/blog/compressionstreams*
434. *https://developer.mozilla.org/docs/Web/API/Clipboard_API*
435. *https://developer.mozilla.org/docs/Web/API/Clipboard_API#security_considerations*

```
// Write text
await navigator.clipboard.writeText("Hello from Web Almanac!");

// Read text
const text = await navigator.clipboard.readText();

console.log(text); // "Hello from Web Almanac!"
```

非同期Clipboard APIはChromiumベースのブラウザ、Safari、Firefoxでサポートされています。ウェブカスタムフォーマットなどのリッチなクリップボードデータのサポートはChromiumベースのブラウザのみです。

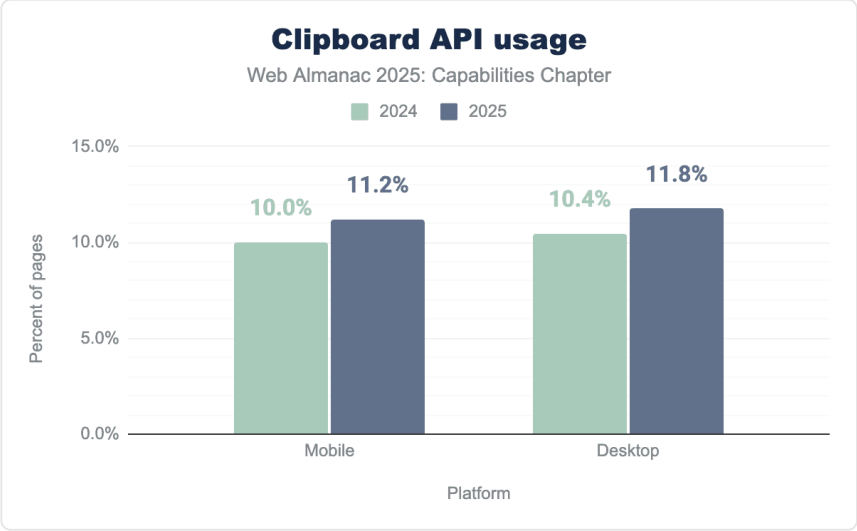


図10.2. Clipboard APIの使用状況（2024年〜2025年）。

Clipboard APIは引き続き着実な成長を見せています。モバイルの採用率は2024年の10.0%から2025年の11.2%に、デスクトップは10.4%から11.8%に増加しました。これは、開発者がレガシーな `execCommand()` のクリップボードハックから離れ、コピーボタンや貼り付けワークフローのために非同期APIを採用する動きが進んでいることを反映しています。前年比の伸びは穏やかで、Clipboard APIが新興のケイパビリティではなく、今や確立されたユーティリティであることを示しています。

Web Share API

Web Share API[436]は、ウェブアプリがデバイスのネイティブ共有メカニズムを呼び出せるようにし、ユーザーがテキスト、URL、ファイルをメッセージング、メール、ソーシャルアプリなどの他のインストール済みアプリと共有できるようにします。

メインメソッドは `navigator.share()` で、共有するデータを持つオブジェクトを受け取ります。オプションの `navigator.canShare()` メソッドは、共有を試みる前に提供されたデータ（特にファイル）が共有可能かどうかを確認するために使用できます。

このAPIはクリックなどのユーザーアクションが必要で、プラットフォームの共有シートを起動し、ユーザーが共有先のアプリを選択できるようにします。

```
const data = {
  title: "Web Almanac 2025",
  text: "Check out the latest edition of the Web Almanac!",
  url: "https://almanac.httparchive.org/en/2025/",
};

if (navigator.canShare && navigator.canShare(data)) {
  try {
    await navigator.share(data);
    console.log("Data shared successfully!");
  } catch (err) {
    console.error("Share failed:", err);
  }
} else {
  console.warn("Sharing not supported on this device.");
}
```

Web Share APIはモダンなChrome、Edge、Safariでサポートされています。Firefoxはフラグによってのみ実装しており、デフォルトでは使用できません。

436. https://developer.mozilla.org/docs/Web/API/Web_Share_API

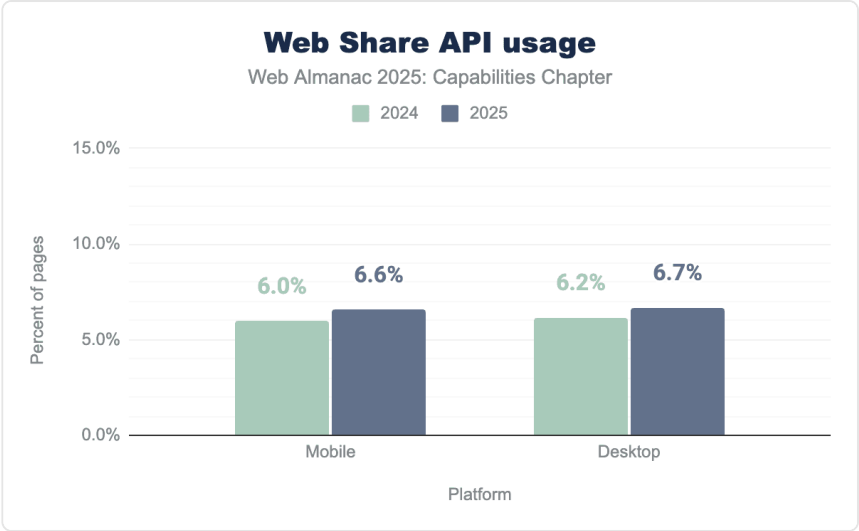


図10.3. Web Share APIの使用状況（2024年〜2025年）。

最も広く採用されているAPIの一つである使用状況にわずかな変化があり、現在最も使用されているAPIのランキングで3位を占めています。

Web Share APIの採用は概ね横ばいで、モバイルは2024年の6.0%から2025年の6.6%へわずかに上昇し、デスクトップは6.2%から6.7%に増加しました。採用はほぼ横ばいながらわずかな上昇を見せています。このAPIは主要ブラウザで成熟と安定の段階に達しており、これらの漸進的な増加は大幅な成長というよりも自然な変動を示しています。

Device Memory API

Device Memory API[437]は、`navigator.deviceMemory`を通じてデバイスのRAMのギガバイト単位の概算値を公開します。これにより開発者は、低メモリデバイスに軽量なページを提供するなど、エクスペリエンスを調整できます。

この値はプライバシー上の理由から丸められた粗い値となっています。

```
console.log(navigator.deviceMemory);
// Example output: 8 (for an 8 GB device)
```

437. https://developer.mozilla.org/docs/Web/API/Device_Memory_API

Chromiumベースのブラウザで利用可能で、SafariやFirefoxではサポートされていません。

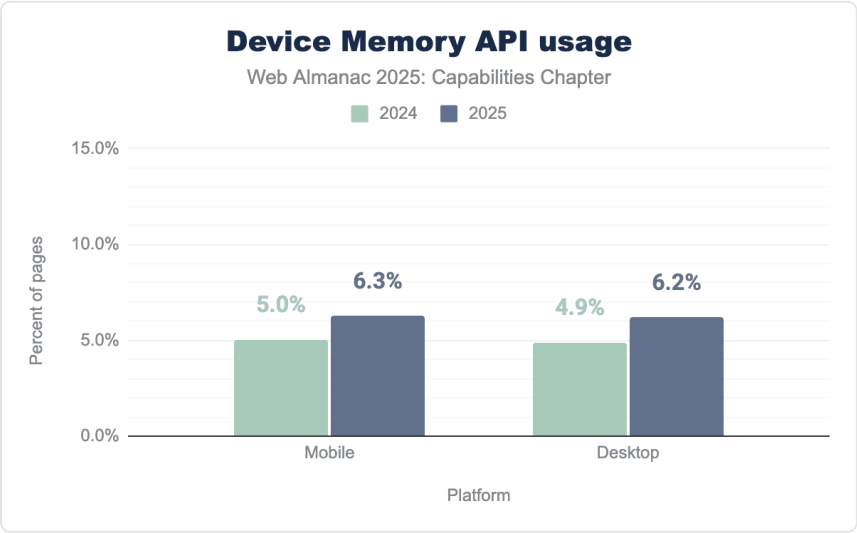


図10.4. Device Memory APIの使用状況（2024年〜2025年）。

Device Memory APIはモバイルで5.0%から6.3%、デスクトップで4.9%から6.2%へと顕著な上昇を見せました。この増加は、開発者が低メモリデバイスに軽量なアセットを提供できるアダプティブパフォーマンス戦略においてAPIの有用性が広く認識されるようになったことを反映しています。もう一つの考えられる説明として、開発者がAIモデルをダウンロードする前に、利用可能なメモリに基づいてデバイスでAI推論を合理的に実行できるかどうかを判断しようとしているという可能性もあります。

より多くの開発者が `navigator.deviceMemory` を活用して低メモリデバイスに軽量なエクスペリエンスを提供しています。採用はChromiumのみの可用性と意図的に粗い値によってまだ制限されていますが、この成長はパフォーマンスを重視するサイトがこれを実際に活用し始めていることを示しています。

Media Session API

Media Session API[438]は、開発者がメディア通知をカスタマイズし、プラットフォームレベルのメディアコントロール（ロック画面、ヘッドセットボタン、スマートディスプレイなど）と統合できるようにします。

`navigator.mediaSession` を使用して、アプリはメディア再生のメタデータとアクション

438. *https://developer.mozilla.org/docs/Web/API/Media_Session_API*

を定義できます。

```
navigator.mediaSession.metadata = new MediaMetadata({
  title: "Web Almanac Podcast",
  artist: "HTTP Archive",
  album: "2025 Edition",
});

navigator.mediaSession.setActionHandler("play", () => {
  audio.play();
});
```

ChromiumベースのブラウザとSafariで広くサポートされています。Firefoxは一部の重要な機能をサポートしていません。

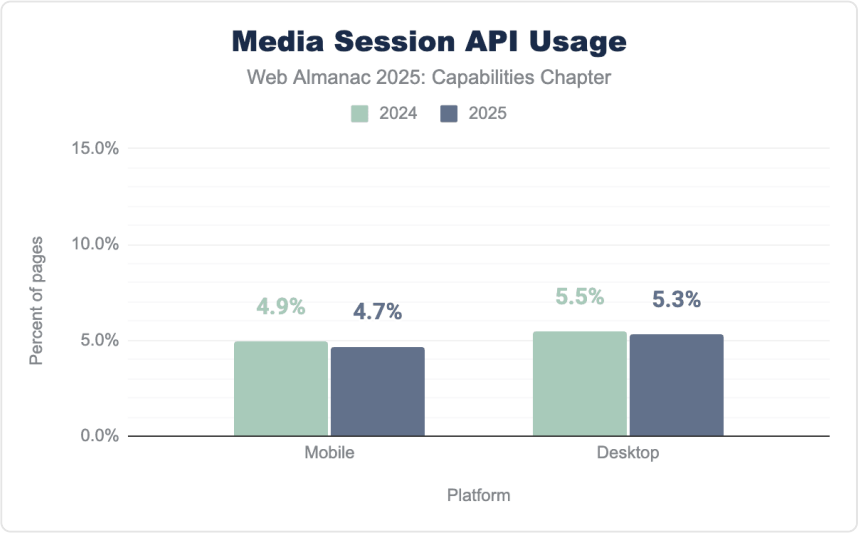


図10.5. Media Session APIの使用状況（2024年〜2025年）。

Media Session APIはわずかな減少を経験しました。モバイルの採用は2024年の4.9%から2025年の4.7%に低下し、デスクトップも5.5%から5.3%にわずかに下落しました。これらの差は軽微であり、意味のある変化よりもデータセットの自然な変動を反映している可能性が高いです。全体的に使用は安定しており、プラットフォームレベルのメディアコントロールとの統合がユーザーエクスペリエンスを向上させる音楽プレーヤーやポッドキャストアプリ

などのオーディオ・ビデオサイトに集中しています。

Add to Home Screen

この機能により、ユーザーはProgressive Web App（PWA）をデバイス上のアプリライクなエクスペリエンスとしてインストール[439]できます。

サイトがインストール可能条件を満たすと、Chromeやほかのブラウザはインストールバッジ（アドレスバーのアイコンや「インストール」メニューオプションなど）を表示し、ユーザーがアプリをホーム画面に追加またはスタンドアロンアプリとしてインストールできるようにします。また、これらの条件を満たさないサイト向けの手動インストールフローもサポートしています。ChromeはさらにAndroidでML駆動のインストールプロンプトを試験的に導入し、ユーザーがインストール可能なエクスペリエンスを発見できるよう支援しています。

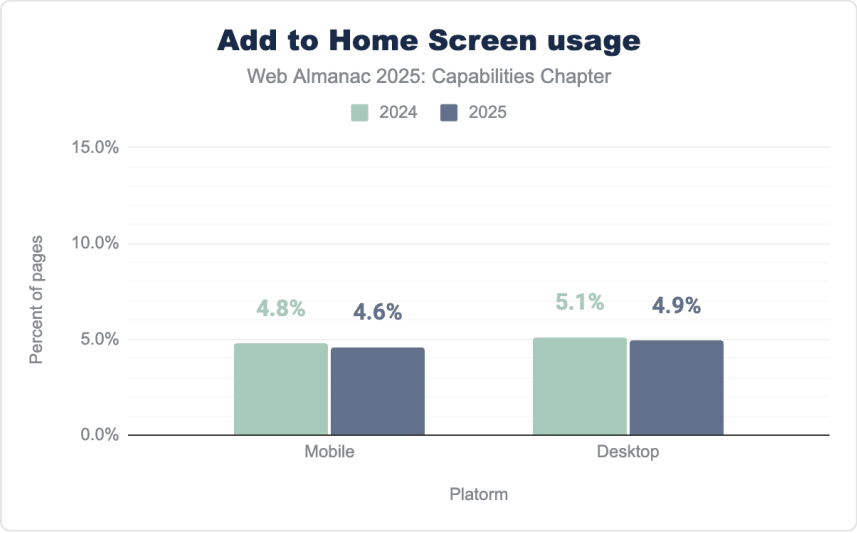


図10.6. Add to Home Screenの使用状況（2024年〜2025年）。

Add to Home Screen機能の採用は横ばいで、モバイルの使用率は2024年の4.8%から2025年の4.6%にわずかに低下し、デスクトップは5.1%から4.9%に低下しました。これらの小さな減少は、実際の下降トレンドよりも通常の変動を反映している可能性が高いです。成長はプラットフォームの断片化によって制約されています。AndroidとChromiumベースのブラウザがインストールプロンプトを公開する一方、iOSは手動のSafari駆動のインストールフローに依存しています。これにより、PWAの採用にもかかわらず広範な普及が制限されています。

439. https://developer.chrome.com/blog/how_chrome_helps_users_install_the_apps_they_value

Media Capabilities API

Media Capabilities API[440]は、ブラウザが指定されたオーディオまたはビデオ設定を効率的にデコードして再生できるかどうかを開発者が照会できるようにします。

アダプティブメディアストリーミングのスムーズさと電力効率に関するインサイトを提供します。

```
const config = {
  type: "file",
  audio: {
    contentType: "audio/mp3",
    channels: 2,
    bitrate: 132700,
    samplerate: 5200,
  },
};

const result = await
navigator.mediaCapabilities.decodingInfo(config);

console.log(result.supported); // true or false
console.log(result.powerEfficient); // true or false
```

広く利用可能で、多くのデバイスとブラウザバージョンで動作します。2020年1月からブラウザ全体で利用可能です。ただし、この機能の一部はSafariなどのブラウザでサポートレベルが異なる場合があります。

440. *https://developer.mozilla.org/docs/Web/API/Media_Capabilities_API*

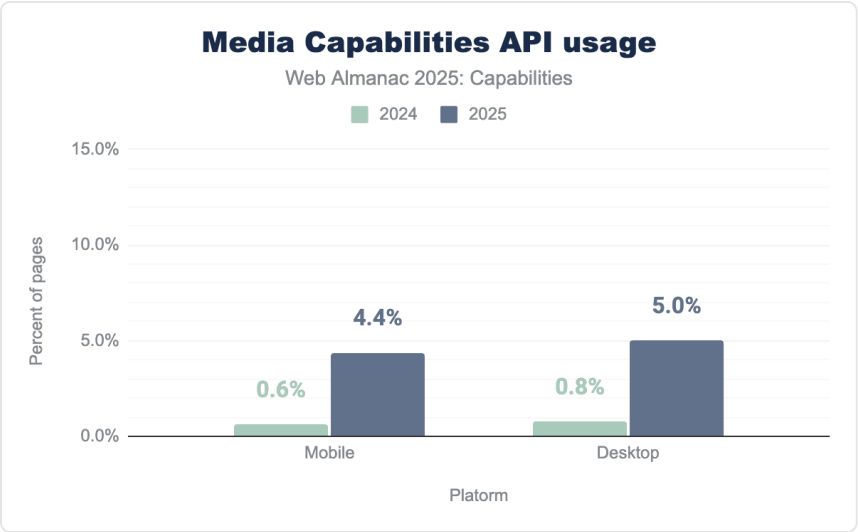


図10.7. Media Capabilities APIの使用状況（2024年〜2025年）。

Media Capabilities APIは過去1年間で劇的な成長を遂げました。モバイルの採用率は2024年のわずか0.61%から2025年の4.37%へ上昇し、デスクトップの使用率は0.75%から5.00%へジャンプしました。この急増は、デバイスに最適なストリームを選択する前にコーデックサポート、再生スムーズさ、電力効率を判断するために `decodingInfo()` を使用するストリーミングプラットフォームによる急速な採用を示しています。漸進的な変化しか見られなかった他のAPIとは異なり、Media Capabilitiesはメディアヘビーなサイトによって牽引される急速な採用軌道に明らかに乗っています。

過去1年間の新機能

2025年のCapabilitiesチャプターで最も注目すべき変化の一つは、ブラウザネイティブのAIおよび言語API[441]が初めて登場したことです。AIはウェブ上で外部サービスやライブラリを通じて長年広く使用されてきましたが、これらのAPIはブラウザが直接提供する組み込みの標準化された言語機能への移行を表しています。

Built-in AI APIs

2025年時点では、実験的なコンテキスト外で利用可能なAPIはサブセットのみです：*LanguageDetector*、*Translator*、*Summarizer*、および *Prompt*（拡張機能に限定）。通常の

441. *https://developer.chrome.com/docs/ai/built-in-apis*

Prompt API、*Writer*、*Rewriter*、*Proofreader* などのその他の組み込みAI機能は実験的なままで、追加のセットアップが必要で、一時的またはトークンベースの制限のもとで動作しています。実験的な機能は本番ウェブサイトに登場する可能性が低いため、この区別は使用データを解釈する際に重要です。

Built-in AI API	デスクトップ	モバイル
LanguageDetector	0.28%	0.26%
Prompt	0.08%	0.09%
Translator	0.28%	0.26%
Summarizer	0.13%	0.14%

図10.8. Built-in AI APIの使用状況

利用可能にもかかわらず、ウェブ全体での使用は非常に限られています。下表に示すように、これらの各APIはデスクトップとモバイルの両データセットでページの1%をはるかに下回る割合でしか登場していません。ただし、実際のサポートは現在ほとんどのデスクトッププラットフォーム（Windows、macOS、Linux、およびChromebook PlusデバイスのChromeOS）に限定されています。Language DetectorとTranslatorが最も多く観測され、それぞれデスクトップページの約0.28%、モバイルページの0.26%で使用されており、PromptとSummarizerはさらに小さな足跡を示しています。

低い採用率は予想されていました。これらのAPIは新しく、多くはまだ進化中であり、現在ChromeとEdgeという限られたブラウザセットでサポートされています。それでも2025年のデータセットへの組み込みは重要な意味を持ちます。HTTP Archiveにブラウザネイティブの組み込みAIプリミティブが初めて測定可能な形で登場したことを示し、今後数年間でウェブにおける組み込みAI機能がどのように進化するかを追跡するためのベースラインを確立しています。

これらのAPIやウェブ上のAIについての詳細な議論は生成AIチャプターも参照してください。

結論

2025年のCapabilities分析は、幅広さと深さの両面で成熟し続けるウェブプラットフォームを示しています。Compression StreamsやAsync Clipboardなどの確立されたAPIは大幅または着実に成長し、より広いクロスエンジンサポートと開発者がレガシーパターンを置き換えていることを反映しています。Web Share、Media Session、Add to Home Screenなどの機能は安定を維持し、年間の変動はわずかにとどまりました。同時に、Media Capabilitiesなどの

スペシャリストAPIはメディアヘビーなサイトで顕著な採用増加を見せており、垂直方向のユースケースでの深い採用を示唆しています。

最も注目すべきは、2025年がLanguageDetector、Translator、Prompt、Summarizerを含むブラウザネイティブのAIおよび言語APIの最初の測定可能な足跡を記録したことです。それぞれがページの1%をはるかに下回る割合でしか登場していなくても、その存在は将来の採用のベースラインを確立し、より高レベルの機能を公開する準備がますます整いつつあるウェブプラットフォームを示唆しています。

今後を見据えると、ブラウザサポートが固まり開発者向けツールが進化するにつれて、継続的な標準化と実際の有用性によって成長が形作られていくでしょう。新しいAPIは実験的な珍しさから、より豊かでスマートなウェブアプリケーションのための実用的な構成要素へと移行していく可能性があります。

著者

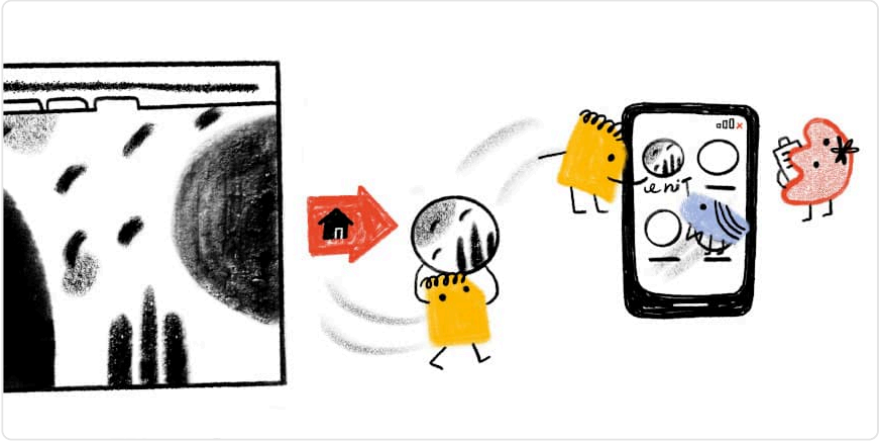
Alba Silvente Fuentes

@dawntraoz Dawntraoz https://www.dawntraoz.com/

Alba Silventeは、Fundaのシニアフロントエンドエンジニアです。自身のブログ[442]でフロントエンド開発、Jamstack、ウェブパフォーマンスについて執筆し、カンファレンスでの登壇、テックポッドキャストの主催、オープンソースコミュニティへの積極的な貢献を行っています。Google Developer Expert（ウェブ技術）、Microsoft MVP、Women Tech Makersアンバサダーでもあります。

442. *https://www.dawntraoz.com/*

部 II 章 11
PWA



Diego Gonzalez と Michael Solati によって書かれた。
Maxim Salnikov、 Kai Hollberg と Aaron Gustafson によってレビュ。
Onur Güler による分析。
Barry Pollard 編集。
Sakae Kotaro によって翻訳された。

はじめに

2015年、私たちは初めてプログレッシブウェブアプリケーションについて読みました[443]。レスポンシブ、接続独立性、アプリライクなインタラクション、常に最新、安全、発見可能、再エンゲージ可能、インストール可能、リンク可能という9つの属性が、当時ウェブ技術で達成できる最先端を定義していました。*PWAのコンセプトは10年前に生まれ*、構想から10年経った今、この技術群の現状を誇りを持って振り返ります。

PWAのコンセプトはこの10年で大きく進化し、異なるブラウザがさまざまなバリエーションと異なる名称でサポートしています。モバイルブラウザで「ホーム画面に追加」を通じてウェブアプリへのアクセスを可能にする方法として始まったアイデアは、今や複数のプラットフォームとデバイスで存在感を示しています。これらの技術群はウェブコンテンツを基盤となるプラットフォームに直接統合し、高度なケイパビリティとよりネイティブに近いルッ

443. *https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/*

ク＆フィールを同時に実現します。ここ数年でPWAにもたらされた変化を見ていきましょう。

PWA/ウェブアプリの変化

ここ数年、ウェブアプリにはさらなるカスタマイズ、アプリケーション動作の高度な制御、パフォーマンス向上を可能にする新機能が追加されてきました。何より、複数のエンジンでウェブアプリのサポートが進展しています！

Chromiumベースのブラウザは、最小要件が満たされるとアプリケーションのインストールを促します。かつてはマニフェストファイル、サービスワーカー、セキュアな接続を持つアプリが対象でした。しばらくの間、これがPWAインストール可能性の「3要素」でした。これが変わり、現在はマニフェストのみが必要です（HTTPS接続は引き続き必要ですが）。サービスワーカーはEdgeやChromeなどのブラウザがインストールプロンプトを表示するための要件ではなくなりました。

一方、Safariはウェブアプリのインストールプロンプトを表示しません。ただし、macOS 14ではDockに追加する[444]ことで任意のウェブページをアプリとしてインストールできます。

PWAにとって大きなニュースとして、Firefoxがバージョン143からウェブアプリをサポートするようになりました[445]！これは現在WindowsのPWAのみで利用可能で、「Mozilla Connectコミュニティからのトップリクエスト」[446]に応えるものです。

ウェブベースのウェブアプリにおいて、サービスワーカーを持たないことへの懸念の一つはオフラインサポートです。サービスワーカーを使用することで、開発者はウェブアプリのUIを構成するリソースをキャッシュして優れたオフライン体験を提供できます。要件の変更（Chromium）またはデフォルト動作の変更（SafariとFirefox）により、ブラウザは接続不足からユーザーを守るデフォルトのオフライン体験を提供するようになっています。

最高のオフラインUXにはサービスワーカーが依然として必要です。ウェブアプリのインストール可能性の最新情報はMDN記事[447]を参照してください。

この概要を踏まえ、データに踏み込んで現在のPWAの状態を理解しましょう。可能な限り、今年のデータを3年前のデータ[448]（最後にPWAチャプターが存在した時）と比較します。

444. *https://support.apple.com/en-us/104996*
445. *https://support.mozilla.org/en-US/kb/web-apps-firefox-windows*
446. *https://connect.mozilla.org/t5/discussions/how-can-firefox-create-the-best-support-for-web-apps-on-the/m-p/60561/highlight/true#M21220*
447. *https://developer.mozilla.org/docs/Web/Progressive_web_apps/Guides/Making_PWAs_installable*
448. *https://almanac.httparchive.org/ja/2022/pwa*

サービスワーカー

サービスワーカーは、バックグラウンド同期、オフラインサポート、プッシュ通知などの高度なケイパビリティをウェブアプリに提供するうえで引き続き不可欠な存在です。今年のデータによると、ウェブプロパティの約5分の1がサービスワーカーを使用しています。

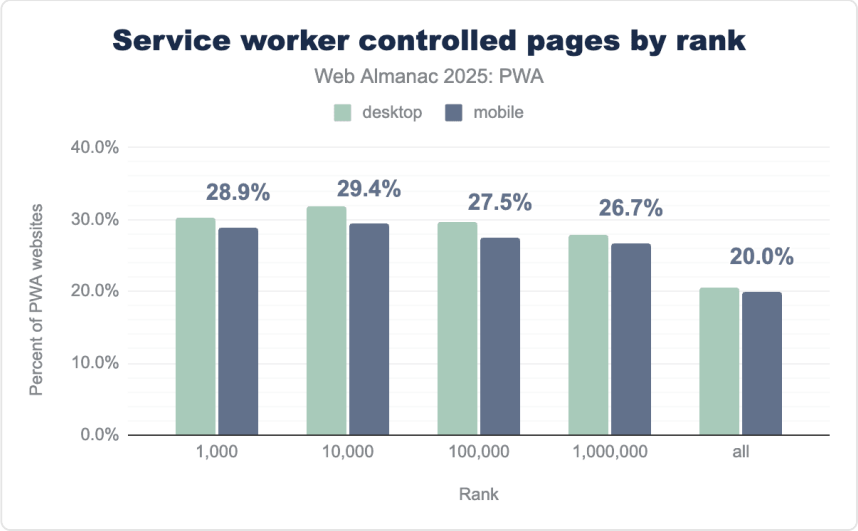


図11.1. ランク別のサービスワーカー制御ページ。

まず、ランク別のサービスワーカー制御ページを見ていきましょう。上位1,000ページでは、30.3%（デスクトップ）と28.9%（モバイル）がサービスワーカーによって管理されています。これは3年前のデータと比べて約20%の増加です。

全体的に、すべてのランクグループで強い増加が見られ、 全PWAウェブサイトの割合は2022年の1.4%（デスクトップとモバイル）から20.5%（デスクトップ）と20.0%（モバイル）へと上昇しました。

以下では、イベント、メソッド、オブジェクト別のサービスワーカーのケイパビリティ使用データを示します。

サービスワーカーのイベント

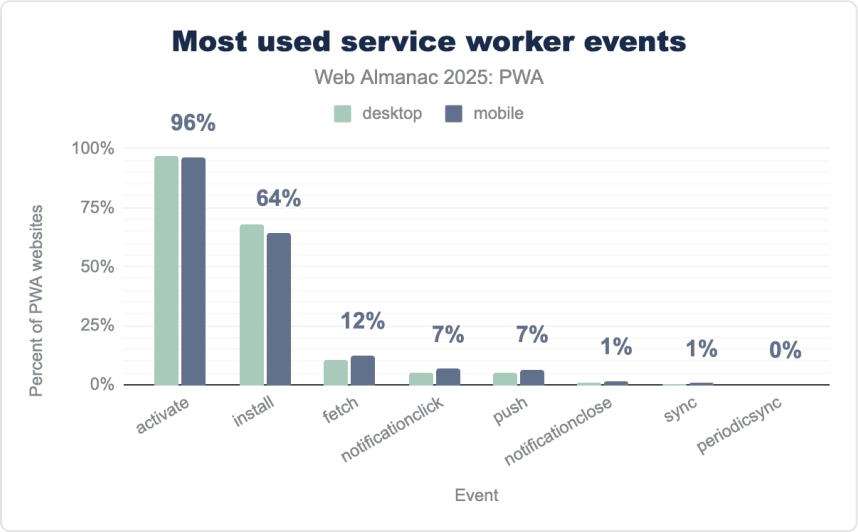


図11.2. もっとも使用されているサービスワーカーのイベント。

サービスワーカーで最も使用されているイベントは`activate`（アクティベーション）で、ほぼすべてのサービスワーカー（PWAの約96%）が使用しています。`install`イベントは約64%の使用率で2位を占めています。`install`と`activate`はともにコアライフサイクルイベントであるため、これらの数字は驚くべきものではありません。アプリケーションがロード時間を短縮するためにリソースをキャッシュし、サービスワーカーの管理を行っていることを示唆しています。

`fetch`、`notificationclick`、`push`などの高度なイベントの使用はかなり低く、これらがネットワークリクエストの傍受、デフォルトのオフラインUXのバイパス、プッシュサービスからの通知配信といったより高度なシナリオに該当するケイパビリティであることが原因と考えられます。

サービスワーカーのメソッド

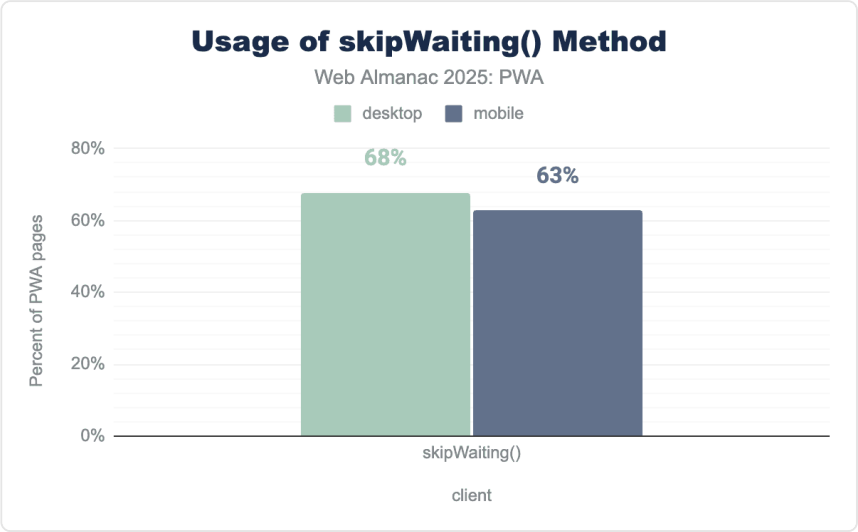


図11.3. もっとも使用されているサービスワーカーのメソッド。

最も使用されているサービスワーカーメソッドを見ると、`skipWaiting()`が顕著な使用率を示しており、デスクトップで68%、モバイルで63%の利用率です。ブラウザはサービスワーカーを即座にアクティブにして古いものと置き換えます。これは開発者がユーザーが古いアセットのままになることを防ぎたいということを意味しています。ダッシュボードやメッセージングアプリのような頻繁なアップデートが必要なアプリケーションに有益で、ユーザーがアプリケーションの最新バージョンをすぐに入手できるよう助けます。

サービスワーカーのオブジェクト

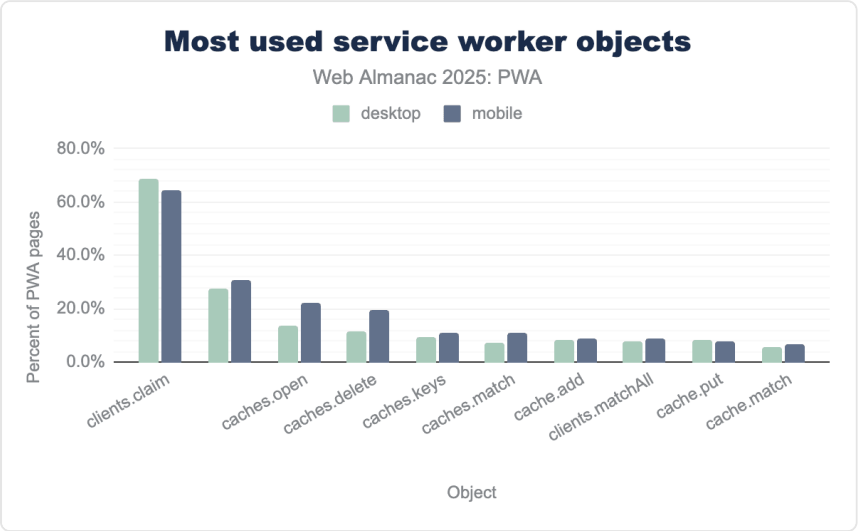


図11.4. もっとも使用されているサービスワーカーのオブジェクト。

最も使用されているサービスワーカーオブジェクトは`clients`、`caches`、`cache`です。`clients`については、`clients.claim()`を呼び出してサービスワーカーがすべての開いているページを制御する方法として期待通りの結果です。`caches`については管理メソッドがトップに登場しており、開発者がアセットを最新の状態に保って高速なページロードを実現するために使用するメソッドであることを考えると、これも驚くべきことではありません。

先に示唆したように、これらの主要なメソッドは`claim`、`open`/`delete`/`keys`/`match`、`add`に対応しています。

登録プロパティ

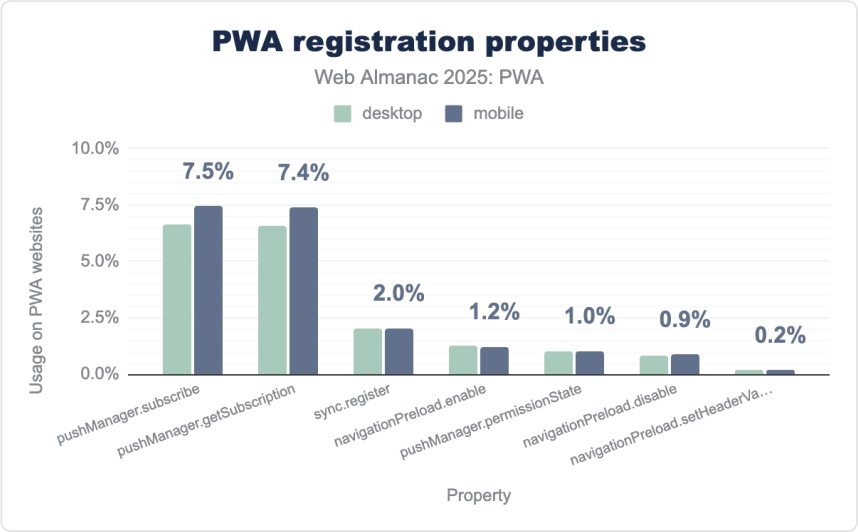


図11.5. もっとも登録されているサービスワーカープロパティ。

サービスワーカーの機能をさらに掘り下げると、データはPWAが使用している高度なケイパビリティに光を当てています。デスクトップとモバイル全体でサービスワーカーを使用しているすべてのPWAのうち、約7%がpushManagerに登録し、2%がsyncに登録し、2%がnavigationPreloadに登録しています。

ウェブアプリマニフェスト

ウェブアプリマニフェストは今や、これまで以上にウェブアプリの最も重要な部分です。ルック＆フィールを定義し、インストール後に利用可能な高度なケイパビリティを有効にし、ウェブアプリケーション全体を識別する不可欠な要素になりつつあります。ただし、有効に機能するためにはマニフェストファイルが正しく形成されている必要があります。

94.9%

図11.6. デスクトップで解析可能なモバイルマニフェストファイルの割合。

今年はデスクトップサイトの94.5%、モバイルサイトの94.9%が解析可能です。前回のデー

タセットから変化はなく[449]、前回同様、マニフェストファイルが解析可能であることは機能の完全性や最低限の利用可能性を意味するわけではありません。マニフェストの多くの値は重要に見えても、適切なフォールバックが用意されています。

これらの解析可能なマニフェストから、個々のフィールドを見ていきましょう。これにより開発者がマニフェストファイルをどのように使用しているか、また2022年以降に変化があったかどうかを理解できます。

マニフェストプロパティ

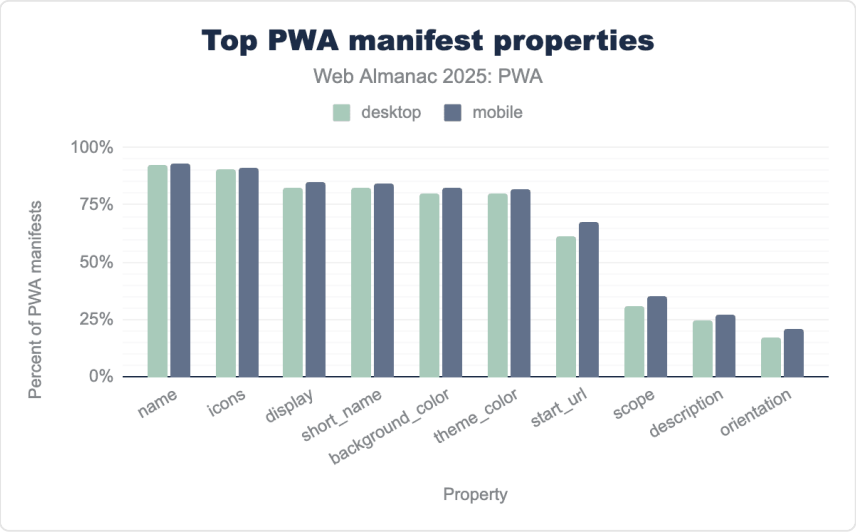


図11.7. もっとも使用されているマニフェストプロパティ。

最もよく使用されているPWAマニフェストプロパティは`name`、`icons`、`short_name`、`display`、`background_color`です。上位4つの最も使用されているプロパティは2022年と同じ[450]で、順序にわずかな変化があります。

Web Almanacがスキャンしたマニフェストファイル全体で個々のメンバーがどのような評価を受けているかを見ていきましょう。特に注記がない限り、値はモバイルとデスクトップで非常に似ているため、両方を合わせて言及します。

449. https://almanac.httparchive.org/ja/en/2022/pwa#fig-9
450. https://almanac.httparchive.org/ja/2022/pwa#マニフェストのプロパティ

マニフェストフィールド	デスクトップ	モバイル
name	92%	93%
icons	90%	91%
short_name	82%	85%
display	82%	85%
background_color	80%	82%
theme_color	80%	82%
start_url	61%	68%
scope	31%	35%
description	25%	27%
orientation	17%	21%

図11.8. マニフェストプロパティ。

`categories`メンバーを指定しているマニフェストでは、トップカテゴリは以下の通りです：

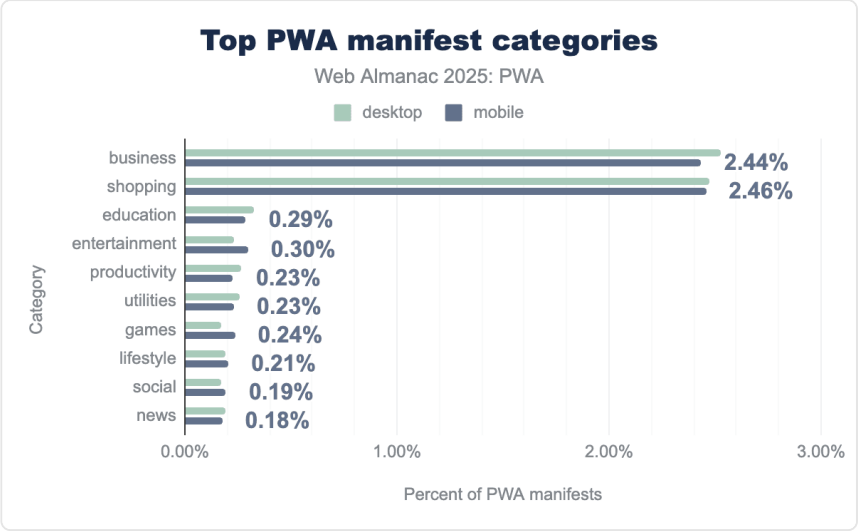


図11.9. もっとも使用されているマニフェストカテゴリ値。

マニフェストキー `categories` は多くのPWAで使用されていませんが、これらの結果はウェブアプリがより人気のあるトップの縦型市場を示唆しています。

マニフェストの `display` 値

displayメンバーはウェブアプリの優先表示モードを指定するために使用されます。ブラウザによってこれらの値の解釈は異なりますが、全体的にブラウザのUXをどれだけ表示/非表示にするかをUAに示します。

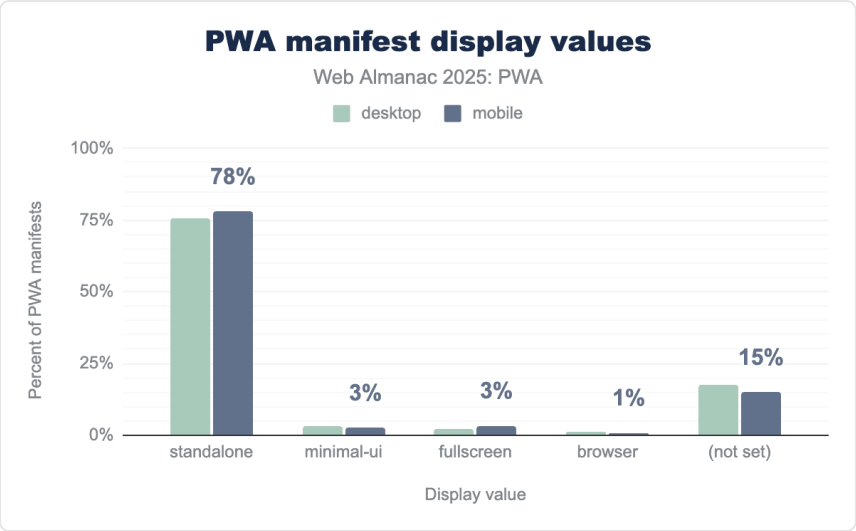


図11.10. マニフェストで最も使用されているdisplay値（standalone）。

ほとんどのウェブアプリ（78%）はdisplayメンバーに `standalone` 値を選択しています。ドキュメントによると、 `standalone` は「スタンドアロンのネイティブアプリのようなルック＆フィールでアプリを開く」とされています。開発者がアプリをよりネイティブに見せたい、ブラウザのクロームや他のUXを取り除きたいというのがウェブアプリの一般的な意図であるため、これは驚くべきことではありません。最終的にはこれは実装によって異なり、例えばChromiumブラウザでは `...` メニューが表示され、Firefoxではインストールされたアプリにも URLバーなどのブラウザUIが残ります。

マニフェストの`icons`サイズ値

図11.11. マニフェストで最も使用されているアイコンサイズ値。

トップサイズは192x192と512x512です。

マニフェストの`orientation`値

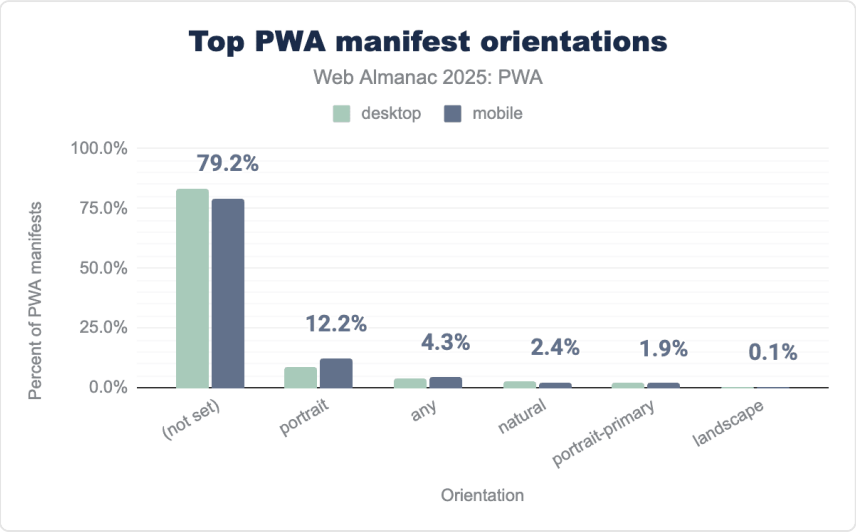


図11.12. マニフェストで最も使用されているorientation値。

当然ながら、PWAの約79.2%はorientationを設定していません。レスポンシブデザインとモダンな開発により、アプリの向きを定義する必要性が低くなっていますが、portraitは約12.2%で2位を占めています。

サービスワーカーとマニフェストの使用状況

最も使用されているサービスワーカーとマニフェストの機能に関する最新データを見てきました。2025年では、サイトの約5分の1がサービスワーカーを使用し、約10分の1がマニフェストを使用していることがわかります。

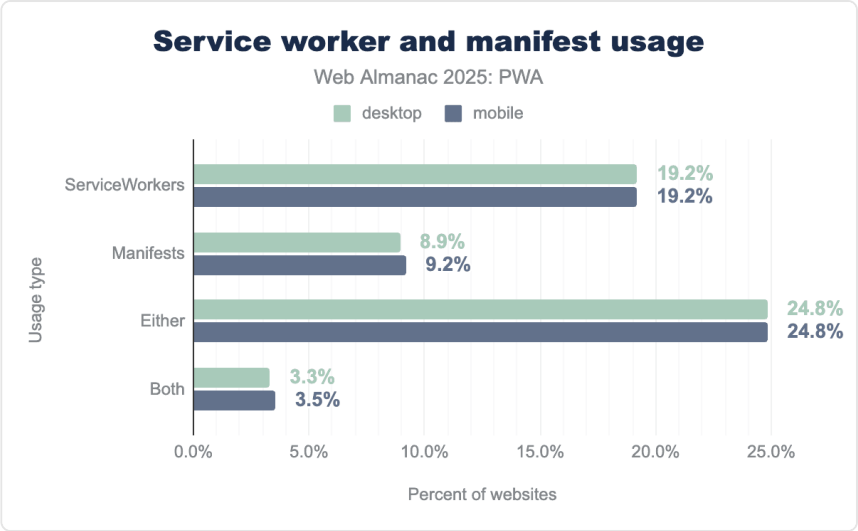


図11.13. PWAサービスワーカーとマニフェストファイルの使用状況。

全体として、2022年版Web Almanac[451]のデータと比べると大きな変化があります。サービスワーカーの使用率は約1.7%から19.2%へと大幅に跳ね上がりました。これは採用率の約10倍増加です。マニフェストの使用率はわずかに高いものの、3年前とほぼ同じ水準に留まっています。

これをさらに掘り下げると、Google Tag Managerがサービスワーカーを有効にしたことが成長の大きな要因となっているようです。

PWAとFugu API

2025年のPWAで使用されている高度なケイパビリティトップ10は以下の通りです。

451. *https://almanac.httparchive.org/ja/2022/pwa#fig-10*

ケイパビリティ	モバイル	デスクトップ
Compression Streams	18.4%	20.9%
Async Clipboard	17.9%	19.1%
Device Memory	10.7%	10.6%
Web Share	9.6%	9.8%
Media Session	6.8%	7.8%
Add to Home Screen	6.8%	7.3%
Media Capabilities	6.3%	7.3%
Cache Storage	9.00%	3.00%
Service Worker	3.7%	3.3%
Push	1.7%	1.6%

図11.14. PWAで使用されている高度なケイパビリティトップ10。

2025年におけるこれらのAPIの採用状況をより深く掘り下げるためのケイパビリティ専用チャプターがあります。

通知とPWA

通知はアプリにとって理にかなった機能で、ユーザーがアプリケーションに再エンゲージできるようにします。これは物議を醸すケイパビリティで、ユーザーに承諾させようとする悪いUXやダークパターンが数多く存在します。データによると、デスクトップとモバイルの両方で、ユーザーが最も一般的にとる行動はこれらのリクエストを無視することです。

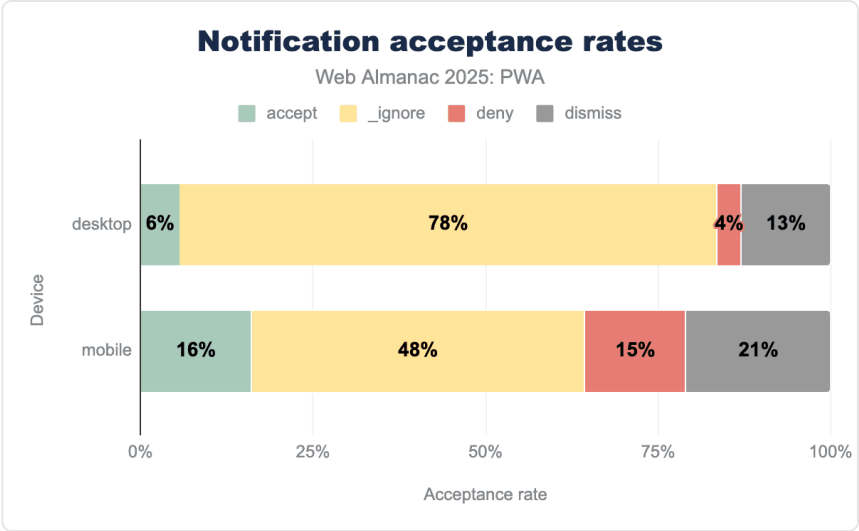


図11.15. PWA通知の承諾率。

デスクトップの通知承諾は圧倒的に無視されており、78%が無視され、モバイルではわずかに低いものの依然として高い48%が無視されます。これはユーザーが日常的に経験している通知疲れを考えると驚くべきことではありません。

結論

PWAが誕生して10年が経ちました。技術とケイパビリティが成熟するにつれてランドスケープは少し変化し、現在はすべての主要ブラウザ（Edge、Chrome、Firefox、Safari）がある程度ウェブアプリをサポートしています。ウェブアプリは引き続き成長を示しており、クロールされたウェブサイトの約4分の1がサービスワーカーまたはマニフェストファイルを持っています。

新機能のペースが遅くなったと感じているなら、それは正しいです。数年前と比べると今日ローンチされるウェブアプリ機能は少なくなっています。これはプラットフォームがケイパビリティの面で、ウェブ技術スタックを使ってさまざまなユースケースを作成できる成熟度に達したという事実と関係していると思います。一方で、クロスエンジンのケイパビリティサポートと密接に結びついた採用の問題もあります。一部の主要ブラウザがいくつかの高度な機能をサポートし始めたのはごく最近のことで、ウェブアプリを取得またはインストールする可能性についてはなおさらです。

実装者のサポートが遅いためと思われる高度なケイパビリティの控えめな使用が続く中、Web Shareなどの古いケイパビリティも登場し始め、さらには`window-controls-`

`overlay` のようなよりニッチなUX機能もデータに現れ始めています。ゆっくりと着実にこれらの機能はコモディティ化されつつあります。

PWAチャプターがあった最後のWeb Almanacを振り返ると、以下のことに気づきます：

- 2つの追加ブラウザ（SafariとFirefox）がウェブアプリをサポートするようになりました。
- サービスワーカー制御ページは2022年からすべてのPWAウェブサイトで約20%増加しました。
- 3年前と比べてサービスワーカーイベントの多様性が少なくなっています。`notificationclick`、`push`、`fetch` を使用しているPWAの割合が減少しています。
- `pushManager` 登録の割合は2022年から低下しており、これは通知のプッシュよりもパフォーマンスに焦点を当てたサービスワーカーの使用（下記注記参照）が増加しているためと考えられます。
- 2022年から2025年の間にPWA技術の使用合計（デスクトップ1,250万、モバイル1,550万）は約2倍になりました（2022年はデスクトップ540万、モバイル790万）。サービスワーカーの使用率は約10倍に急増し、マニフェストは8〜9%の使用率とほぼ同水準を維持しています。
- 通知プロンプトを無視することが依然としてユーザーにとって最も一般的な行動です。

これらはパーセンテージであるため、「呼び出しの多様性の低下」や「登録の減少」は特に拒否や不使用のサインではなく、コア技術の使用が増加した結果であることに注意する価値があります。例えば、PWA登録プロパティにおける `pushManager.getSubscription` の合計は、過去3年間で52,000ページから約300万ページ未満へと増加しています。

2025年版Web AlmanacのPWAチャプターを締めくくるにあたり、インストール機能をプラットフォームに直接組み込むことでウェブアプリの配布を民主化する方法を模索する取り組みと合意が進行中です。次の10年間でPWA技術がレスポンシブデザインと同じ運命をたどり、コモディティ化されて、アプリケーションライフサイクル管理に優れたUXを持つウェブアプリが、ウェブアプリの可能性を再定義する新しいケイパビリティとともに標準になることを願っています。ウェブアプリの次の10年に乾杯！

著者

Diego Gonzalez

@diekus diekus https://diek.us

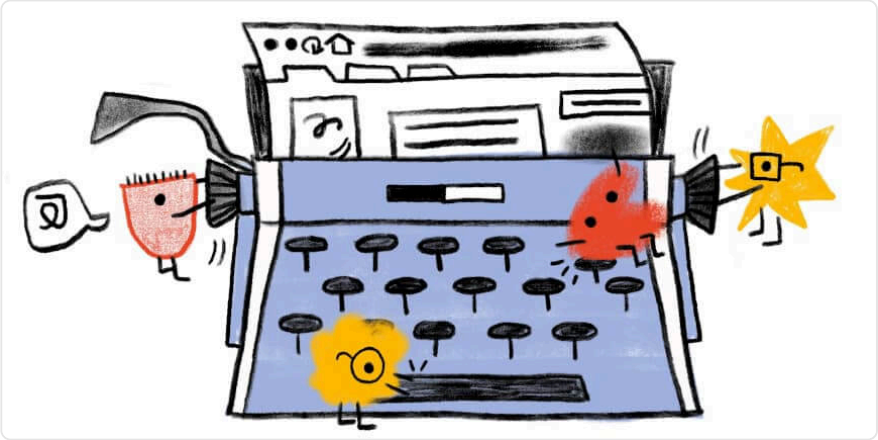
Diego GonzalezはMicrosoft Edgeブラウザのウェブプラットフォーム機能に取り組むコスタリカ出身のエンジニアです。

Michael Solati

@MichaelSolati MichaelSolati https://michaelsolati.com

MichaelはAmplicationのデベロッパーアドボケートで、開発者がAPIを構築するのを支援することとIPAを飲むことに注力しています。また、Web GDEでもあり、ウェブ上での魅力的なエクスペリエンスの創造とウェブの神秘的な世界に情熱を見出しています。

部 III 章 12
CMS



Vahe Arabian によって書かれた。
Dan Knauss によってレビュ。
Onur Güler と Alon Kochba による分析。
Samuel Fatola と Sreemoyee Bhattacharya 編集。
Sakae Kotaro によって翻訳された。

はじめに

コンテンツ管理システム（CMS）は今やほぼすべてのウェブサイトを支えています。2025年までに、CMSは単にページを作成・公開できるようにするだけでなく、はるかに多くのことを行うようになりました。サイトの構築方法、使いやすさ、検索エンジンやAIツールへのコンテンツの露出方法を形作っています。サイトが大規模化し、より活発でパーソナライズされるにつれ、CMSは高速で直感的かつ信頼性の高いサイトと、遅くフラストレーティングなサイトの違いを生む存在となっています。本チャプターはHTTP Archiveのデータを使用してCMSのランドスケープを調査します。

CMSを使用しているサイト数、上位サイトがウェブ全体とどう異なるか、CMS搭載サイトのパフォーマンス、そしてこれらのプラットフォームが大規模に運用される中で台頭している新しいケイパビリティを検討します。CMS機能リストを比較するだけでなく、本チャプターではCMSのデフォルト設定、ホスティングモデル、ビルドアプローチが、上位100万サイトと上位1万サイトの双方において、ユーザーと検索エンジンにとってのウェブの実際のエク

スペリエンスをどのように形成するかに焦点を当てています。

データによると、CMSはサイトの大多数で使用されており、CMSを使用していないサイトは年々減少しています。これが速度、使いやすさ、発見可能性にもたらすトレードオフを理解するために、本チャプターではまずCMS全体の採用状況を探り、次にパフォーマンスとユーザーエクスペリエンスに何を意味するかを検討し、最後にCMSの普及とともに登場している新しいツールとパターンを見ていきます。

CMSとは？

コンテンツ管理システムとは、ウェブ上でコンテンツを作成・更新するたびにコードを編集することなく行えるようにするソフトウェアです。ほとんどのCMSはコンテンツとプレゼンテーションを分離しており、編集者はHTML、CSS、JavaScriptを書かずにテキスト、画像、構造を変更できます。開発者はテーマ、プラグイン、モジュールを構築してCMSを拡張できます。

大きく分けると、CMSには2つの主要なタイプがあります。クラシックなモノリシックCMSは、コンテンツの保存、プレゼンテーション、デリバリーを1つのシステムに統合します。

ヘッドレスまたはコンポーザブルCMSは、コンテンツ管理をそれを表示するサイトやアプリから分離します。実際には、ほとんどのCMSはコンテンツのモデリング、編集、メディア管理、機能拡張、他システムとの統合という共通のタスクセットを持っています。これらのタスクへのアプローチの違いは、ウェブ全体の採用および実装パターンを見ると明確になります。

CMSの導入

2025年までに、CMSの導入状況は成熟しつつも専門化が進むウェブを反映しています。ほぼすべてのサイトが何らかのCMSで動作するようになりましたが、これらのプラットフォームの使用方法は地域、トラフィックレベル、基盤技術によって大きく異なります。一つの標準的なシステムに統一されるのではなく、市場は明確に断片化しており、世界のさまざまな地域、ユースケース、スケールで異なるプラットフォームがリードしています。

全体的な採用状況

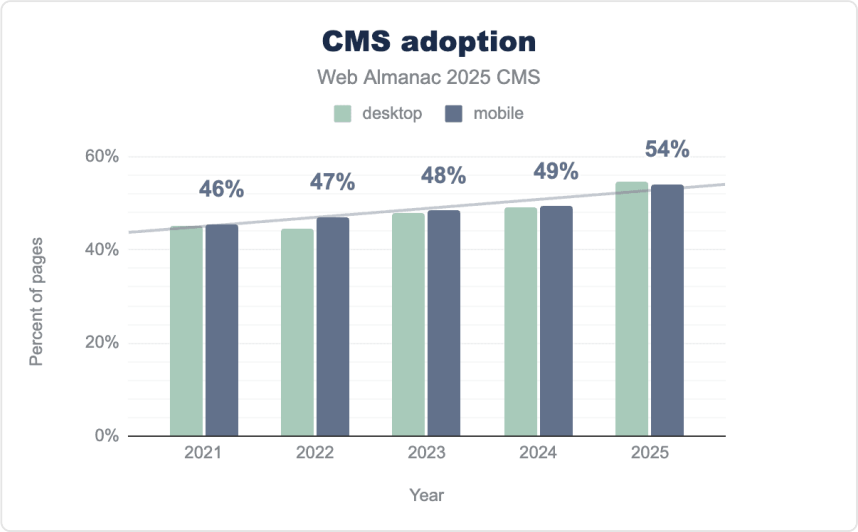


図12.1. CMSの採用状況。

HTTP Archiveの全体的な採用測定では、CMS主導のサイトが2025年に観測されたウェブサイトの54%以上を占めており、CMSがウェブのデフォルトインフラとしての地位を強化しています。

地域別の採用状況

CMSの採用状況は地域によって異なります。

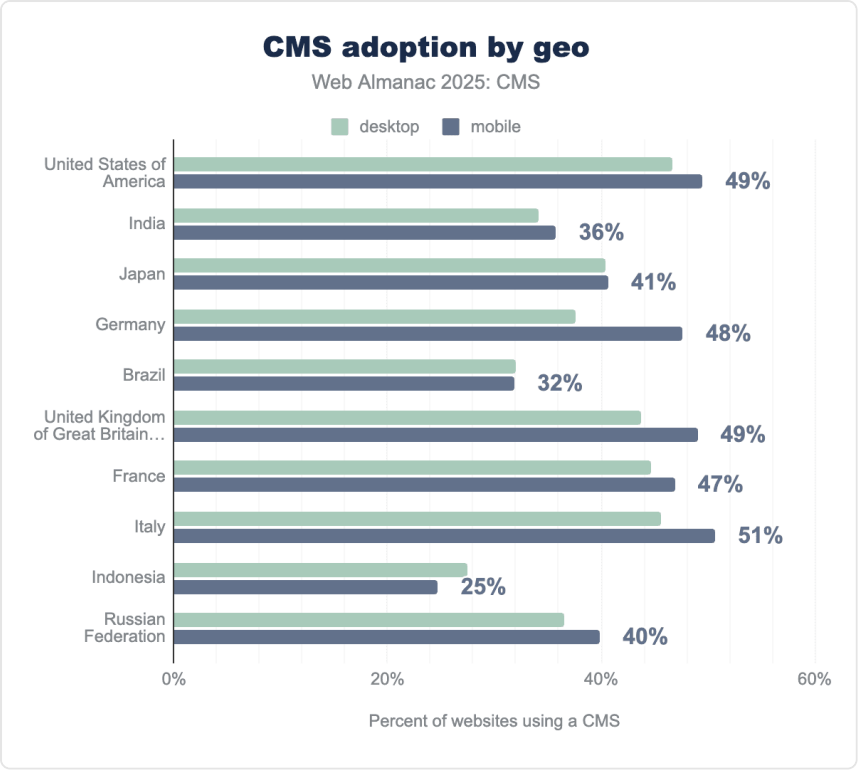


図12.2. 地域別CMS採用状況。

HTTP Archiveのデータは、北米と西ヨーロッパでホスト型CMSプラットフォームの集中度が高いことを示しています。対照的に、オープンソースシステムはアジアと東ヨーロッパの一部で比較的強いポジションを保っています。

ランク別の採用状況

ウェブサイトのランクはCMS採用パターンを形作り続けています。高トラフィックサイトは複雑なワークフロー、深いカスタマイズ、長期的なスケーラビリティをサポートするプラットフォームを選ぶ傾向があります。低トラフィックサイトは運用オーバーヘッドを軽減するホスト型またはオールインワンソリューションを使用する可能性が高いです。

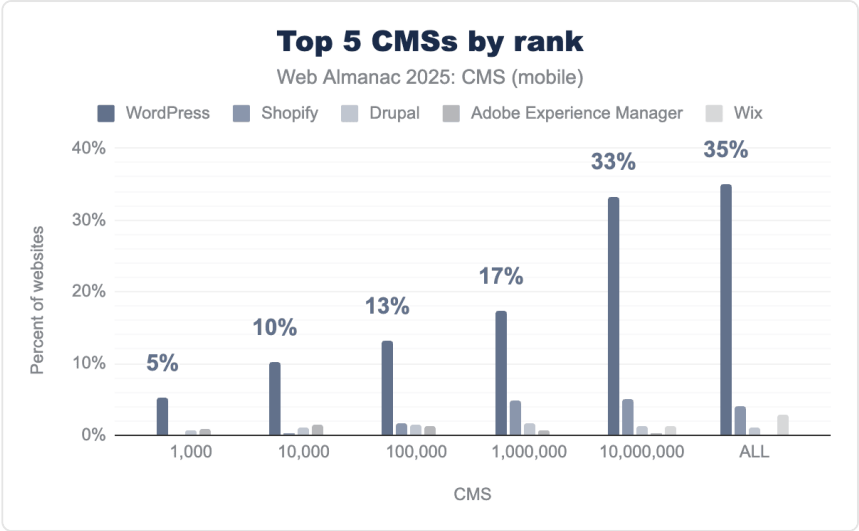


図12.3. ランク別上位5CMS。

上位10,000ウェブサイトの中で、WordPressはCMS使用率の約58%を占め、Drupalは約6〜7%を占めており、全体の市場シェア1%よりはるかに高い比率です。WixやShopifyなどのプラットフォームはこのトラフィックレベルではほとんど存在感がありません。

もっとも人気のあるCMS

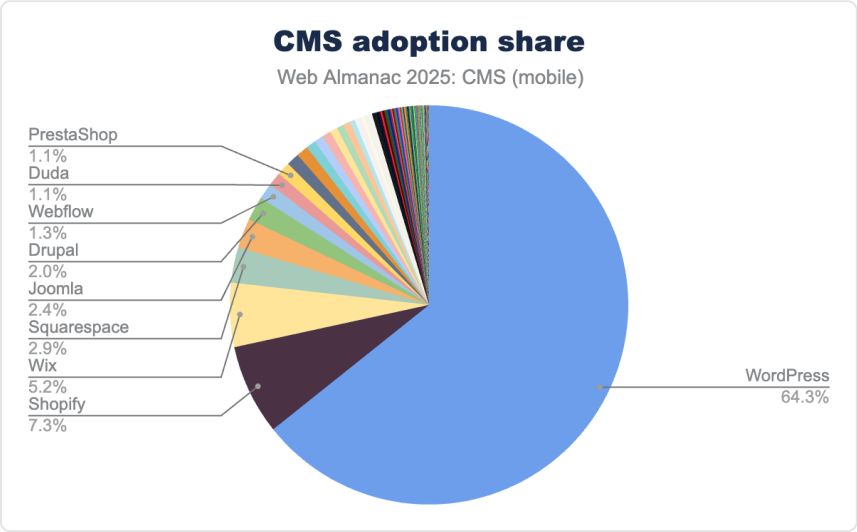


図12.4. CMS採用シェア。

WordPressは2025年も支配的なCMSであり続け、CMS主導サイトの約64%を占めています。ただし、その成長は前年比で1パーセントポイント未満に鈍化しており、単一の競合他社からの大きな脅威よりも市場の飽和を示しています。

データによると、この鈍化は1つのプラットフォームがWordPressを置き換えることによって引き起こされているわけではありません。代わりに、小さな伸びが複数のCMSに分散しています。Shopifyはほぼ7.3〜7.8%のCMSシェアに成長し、Wixは約5%、Squarespaceは約3%になりました。どのプラットフォームの成長も、WordPressの拡大鈍化を相殺するほど大きくはなく、統合よりも断片化したエコシステムを示唆しています。

全体的に、データは市場の成熟とより多くの選択肢によって形成された、より多様なCMSランドスケープを示しています。WordPressは広く採用され続けていますが、新たな成長は単一のデフォルトCMSに集中するのではなく、複数のプラットフォームに分散するようになっています。

急成長するCMS

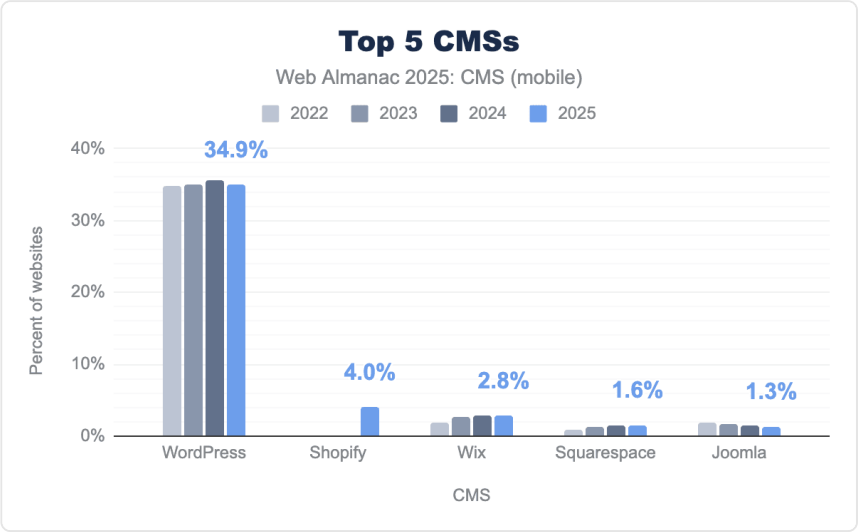


図12.5. 上位5CMS。

2025年のCMSプラットフォーム間の前年比成長は控えめで不均一です。Shopifyは今年の新規エントリーで、これまでEコマースカテゴリのみに分類されていましたが、現在はCMSにも含まれています。Squarespaceは約0.2〜0.3ポイントのわずかながらもポジティブな成長を示し、約3.0〜3.3%に達しています。Wixの成長は前年と比べて大幅に鈍化し、急速な拡大期の後、現在は約5.2%で実質的に横ばいになっています。これらの伸びは漸進的に分散しており、爆発的な成長の新たな波を示すものではありません。

対照的に、WordPressのシェアは2024年のピークから前年比でわずか（1パーセントポイント未満）に低下しており、数十年の拡大後の最初の持続的な鈍化を示しています。それでもWordPressは依然としてCMS主導サイトの大半を支えており、ウェブ全体の明確なリーダーであり続けています。JoomlaやDrupalなどの伝統的なオープンソースプラットフォームは全体シェアの長期的な低下が続いていますが、Drupalは高トラフィックサイトでは依然として不釣り合いに高く代表されています。要するに、2025年に最も急成長しているCMSは特定のニッチで伸びを上げており、WordPressは絶対的な規模では依然として支配的で、より広いエコシステムの基盤であり続けています。

2025年のWordPress

CMSの世界でWordPressが支配的な存在であることを考えると、もう少し詳しく議論する価値があります。

市場とエコシステム

CMSエコシステムにおけるWordPressの重要性は、今や純粋な成長よりも影響力にあります。その圧倒的な規模は、アーキテクチャの選択、パフォーマンスパターン、実際の実装上のトレードオフを理解するための基準点となっています。WordPressはほぼすべてのトラフィック層とユースケースで使用されているため、本番環境での動作はウェブ全体のパフォーマンスとCMS主導の結果に大きな影響を与えています。

この影響力は高度に拡張可能なエコシステムによって支えられています。単一の開発またはホスティングモデルを強制するのではなく、WordPressはプラグイン、テーマ、統合レイヤーを通じて多くのアプローチをサポートしています。この柔軟性により、サイトはプラットフォームを移行することなく、新機能、収益化モデル、編集の複雑さを追加しながら段階的に進化できます。その結果、他のプラットフォームがより狭いセグメントで成長する中でも、WordPressはサイトのライフサイクルの複数の世代にわたって使用され続けることが多いです。

しかし、同じオープン性が重大な変動をもたらします。WordPressサイトは、より緊密に統合されたCMSと比較して、アーキテクチャ、パフォーマンスプロファイル、運用の複雑さの幅広い範囲を示しています。その優位性は均一な結果をもたらさず、代わりに実装の選択によって形成される広い分布を生み出します。次のセクションでは、この柔軟性が実際にどのように現れるか、アーキテクチャ、パフォーマンス指標、スケール時の制約の観点から検討します。

技術的なパフォーマンスと改善

最近のWordPress開発は、サイトがサイズ、編集の複雑さ、ブロックの使用量で成長するにつれて現れるパフォーマンスのボトルネックを削減することに焦点を当てています。インタラクションモデルを再定義するのではなく、コアエンジニアリングの作業は既存のシステムをより高速で、よりキャッシュフレンドリーで、設定の違いに対してより敏感でないようにすることを重視しています。

進歩の主要な領域はエディターのパフォーマンスです。テンプレートの読み込み、ブロックのレンダリング、パターンの再利用の最適化により、エディターの起動と操作の遅延が削減され、ブロックの多いセットアップでテンプレートの読み込みが最大35%速くなりました。再利用可能なブロックパターンとグローバルスタイルの永続的なキャッシュにより繰り返しの計算が削減され、複雑なレイアウトを持つ大規模なサイトの長年のスケーラビリティの問題に対処しています。

メディア処理も改善されました。画像処理パイプラインの更新により、約20%速いAVIF画像生成[452]、より優れた自動サイジング、より信頼性の高い遅延読み込みが実現しました。これ

452. *http://core.trac.wordpress.org/ticket/*

らの変更により、個々のサイトレベルのチューニングを必要とせずに、サーバー側の処理時間とフロントエンドのレイアウトシフトが削減され、読み込みパフォーマンスが向上します。

フロントエンドでは、パフォーマンス作業は不要なペイロードと冗長な処理の削減に集中しています。コアはブロックスタイルとスクリプトをより選択的に読み込み、非表示ブロックのアセットを回避し、生成されたスタイルをより積極的にキャッシュします。プリフェッチやプリレンダリングなどの投機的読み込み技術により、対応ブラウザでの体感速度とLargest Contentful Paintがさらに向上します。総合的に、これらの最適化は一般的な実際のボトルネックを対象とし、さまざまな設定全体で一貫性を向上させています。HTTP Archiveのデータはこのパターンを支持しており、WordPressのパフォーマンスの変動はコアの制限よりも設定によって促進されることを示しています。

アクセシビリティの改善はパフォーマンス作業と並行して進められています。セマンティックマークアップ、キーボードナビゲーション、ラベリング、エディターの使いやすさの強化により、アクセシビリティをテーマやプラグインの選択に完全に依存するものではなく、WordPress出力の基本的なプロパティにすることを目指しています。

2025年のWordPressの進化

これらの変更は、WordPressが拡大への焦点から安定化へとシフトしていることを示唆しています。エディターの応答性、キャッシュの再利用、アセットの削減、変動制御への重点は、WordPressサイトが上位または下位にクラスター化するのではなく、広いパフォーマンス範囲にわたることを示すHTTP Archiveの調査結果と一致しています。コアの変更は、最もパフォーマンスの低い実装を改善し、適切にチューニングされたサイトと不適切にチューニングされたサイトのギャップを狭めることにますます焦点を当てています。

ブロックベースのアーキテクチャとフルサイト編集（FSE）への継続的な投資は、実験というより長期的なコミットメントのように見えます。ブロックモデルから後退するのではなく、WordPressは着実な最適化によってその運用コストを吸収しています。ブロックは現在、コアインフラとして扱われており、パフォーマンス作業は大規模で長寿命のサイトに対してブロックを実行可能にすることに焦点を当てています。

同様に重要なのは、WordPressコアが行おうとしないことです。より広いエコシステムでAI、コラボレーションツール、高度なワークフローの実験が広まっているにもかかわらず、コアは意見の込もったエンドユーザー機能をバンドルするのではなく、API、プリミティブ、後方互換性を優先し続けています。これはWordPressの歴史的なパターンに従っています：エコシステムの端でイノベーションを起こさせながら、コアを保守的で予測可能に保つ。

61758?st_source=ai_mode#:~:text=AVIF%20generation%20has%20improved%20from,:%2095.81%20MB%20(100463863%20bytes)

より二極化しつつあるCMS市場において、これらの選択はWordPressの耐久性のあるウェブインフラとしての位置を強化します。他のプラットフォームが緊密に管理された垂直統合型のエクスペリエンスによって差別化を図る中、WordPressは適応性、長期性、スケールでのリスク削減を引き続き優先しています。2025年のWordPressの技術的な軌跡は、競合他社と機能ごとに追いかけることよりも、ウェブの最も広く多様なスライスにわたって動作可能であり続けることにあります。

ページビルダー

ページビルダーはWordPressエコシステム内の支配的なインターフェースレイヤーになっています。推定によると、WordPressサイトの約60%がページビルダーを使用しており、より速いイテレーション、開発者への依存度の低減、より大きな編集の自律性への需要を反映しています。Elementorが最大の観測されたフットプリントを持ち、WPBakeryとDiviがそれに続き、ネイティブブロックエディターも広く使用されています。

ページビルダーはワークフローを加速するため魅力的です。WYSIWYGの編集、再利用可能なコンポーネント、テンプレートライブラリにより、技術的でないユーザーはコードを書かずにレイアウトを調整できます。これはより広いノーコードおよびコンポーネントベースの開発トレンドと一致しています。多くの組織にとって、これはボトルネックを縮小し、イテレーションサイクルを短縮します。

これらの利点にはトレードオフが伴います。ページビルダーは多くの場合、より複雑なDOM構造とより大きなCSSおよびJavaScriptバンドルを生成し、スケールでのパフォーマンスリスクを高めます。特に古いビルダーは、長期的なメンテナンスとロックインの問題を引き起こしています。パフォーマンスへの期待が高まるにつれ、これらのコストはますます目に見えるようになっています。

これに対応して、主要なビルダーはWordPressコアの規約に近づいています。新しい取り組みは、ブロックネイティブな出力、ショートコードへの依存度の低減、ブロックエディターとFSEとの深い統合を優先しています。コアを置き換えようとするのではなく、多くのビルダーは共有プリミティブの周りに収束し、主にUXレイヤーで競争しています。

ページビルダーは、WordPressのページがどのように組み立てられレンダリングされるかを変え、パフォーマンスに明確な影響をもたらします。レイアウトとデザインをビジュアルコンポーネントに抽象化することで、カスタムコードの必要性を減らしてサイト作成を加速します。しかし、この余分な抽象化はDOMの複雑さ、アセット読み込みの動作、ランタイムの実行にも影響し、ページビルダーをWordPressサイト間のパフォーマンス変動の主要な要因にしています。

歴史的に、多くのページビルダーは重いマークアップと大きなCSSおよびJavaScriptペイロードに関連しており、これはしばしばより遅い読み込みとより弱いCore Web Vitalsをもたらしていました。最近のビルダーは、条件付きアセット読み込み、ミニフィケーション、ショー

トコードの使用削減を追加することでこれを改善しようとしています。これらの取り組みは役立ちますが、ビルダーが多いサイトとより緊密に制御されたビルドの間のパフォーマンスギャップを完全に解消するわけではありません。

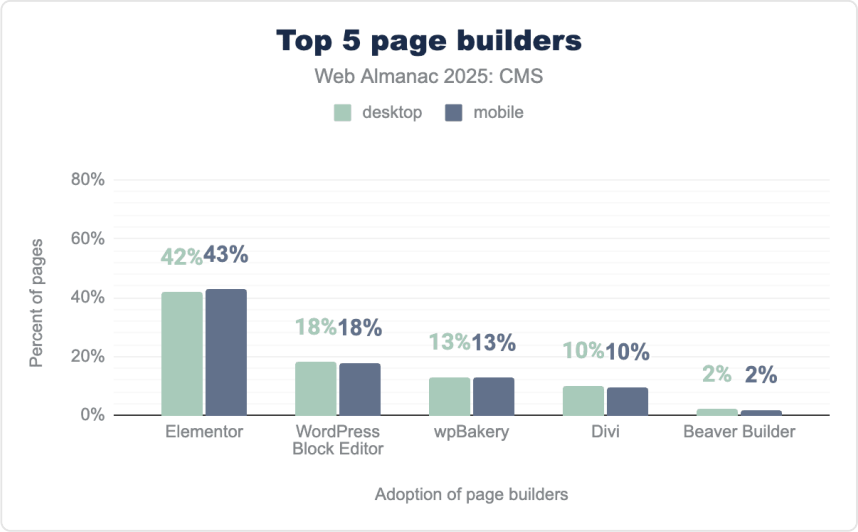


図12.6. 上位5ページビルダー。

使用パターンも変化しています。2024年から2025年の間に、Elementorは最も広く使用されているページビルダーであり続けましたが、そのシェアは約56%から43%に落ち、より断片化したエコシステムを示しています。WordPress Block Editorは約18%に成長し、WPBakeryは約21%から13%に落ちました。Diviは約14%から10%に低下し、Beaver Builderは約2%という小さいながらも安定したシェアを維持しました。これらのトレンドは、古いビルダーからブロックネイティブまたはよりパフォーマンス重視のアプローチへの段階的な移行を示しています。

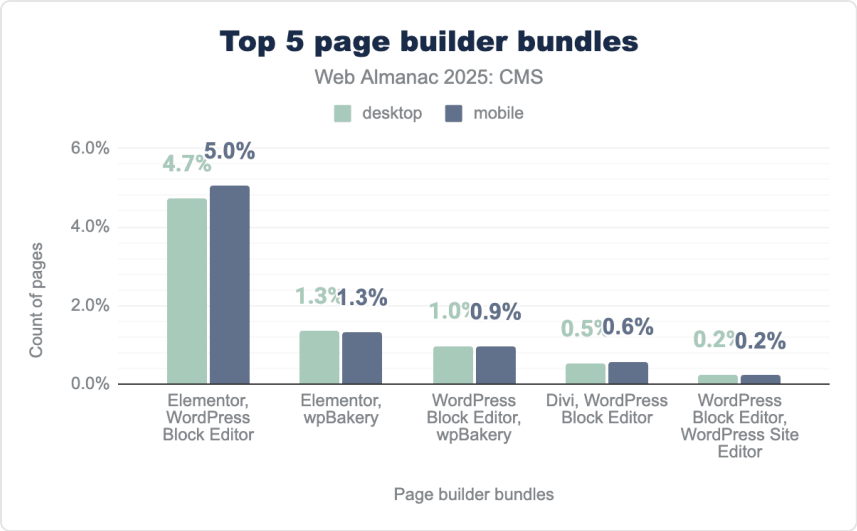


図12.7. 上位5ページビルダーバンドル。

全体的に、データはページビルダーがまだWordPressエコシステムのコア部分であることを示していますが、そのパフォーマンスへの影響はWordPressコアとどれだけ密接に一致しているかに大きく依存するようになっています。マークアップのオーバーヘッドを削減し、グローバルアセットの読み込みを回避し、ブロックエディターと深く統合するビルダーはパフォーマンスコストをより効果的に制限する傾向があります。期待が高まり続けるにつれ、ビルダー間の技術的な差異はビジュアル機能セットよりも重要になる可能性があります。

CMSのユーザーエクスペリエンス

これまでに説明したアーキテクチャとパフォーマンスのトレードオフは、プラットフォームの設計だけでなく、CMSが実際にどのように使用されるかによっても形成されています。CMSのユーザーエクスペリエンス、特に編集者と管理者にとっては、実際の結果の重要でありながらしばしば十分に探求されていない要因です。

プラットフォーム全体において、編集上のUXは構造よりも柔軟性を重視する傾向があり、多くのオプションを提供しながらもガードレールが少ないです。効果的なCMS UXは代わりに、構造化されたコンポーネント、適切なデフォルト、ロールベースの権限を活用して認知的負荷を軽減し、一貫した公開をサポートします。サイトが成長するにつれ、ガバナンス、承認ワークフロー、タクソノミー管理、ローカライゼーションなどの追加ニーズにより、CMS UXはデザインの好みから運用上の要件に変わります。

ほとんどの訪問者には見えませんが、CMS UXはフロントエンドの品質に直接影響します。

制約が少ない編集ツールは、レイアウトの不一致、重いページ、アクセシビリティの問題、古いコンテンツをもたらす可能性があります。このように、バックエンドUXはエンドユーザーのパフォーマンス、発見可能性、アクセシビリティに間接的に影響します。

これらの課題への一般的な対応の一つは、ページビルダーの採用です。ページビルダーは編集者フレンドリーなビジュアルインターフェースを通じてレイアウトとデザインを簡素化することを目指しています。その影響は、WordPressサイトのパフォーマンス変動がコア自体よりも設定とツールの選択によって促進されることを示すHTTP Archiveの調査結果に反映されています。

Core Web Vitals

Core Web Vitalsは、CMSプラットフォームが実際の条件下でどのように機能するかを確認するための実用的な方法を提供します。理論的なベストケースのパフォーマンスを示すのではなく、これらの指標は実際のユーザーが体験するプラットフォームのデフォルト、ホスティングの選択、テーマ、プラグイン、ページビルダーの組み合わせた影響を捉えます。

このセクションでは、主要なCMSプラットフォーム全体のCore Web Vitalsのパフォーマンスを前年比で見ます。制約が厳しく差異が見えやすいモバイルに焦点を当てます。目標は単一の「最速」CMSを選ぶことではなく、プラットフォームコントロールのレベルとエコシステムの複雑さがスケールでのパフォーマンスにどのように影響するかを理解することです。

前年比トレンド

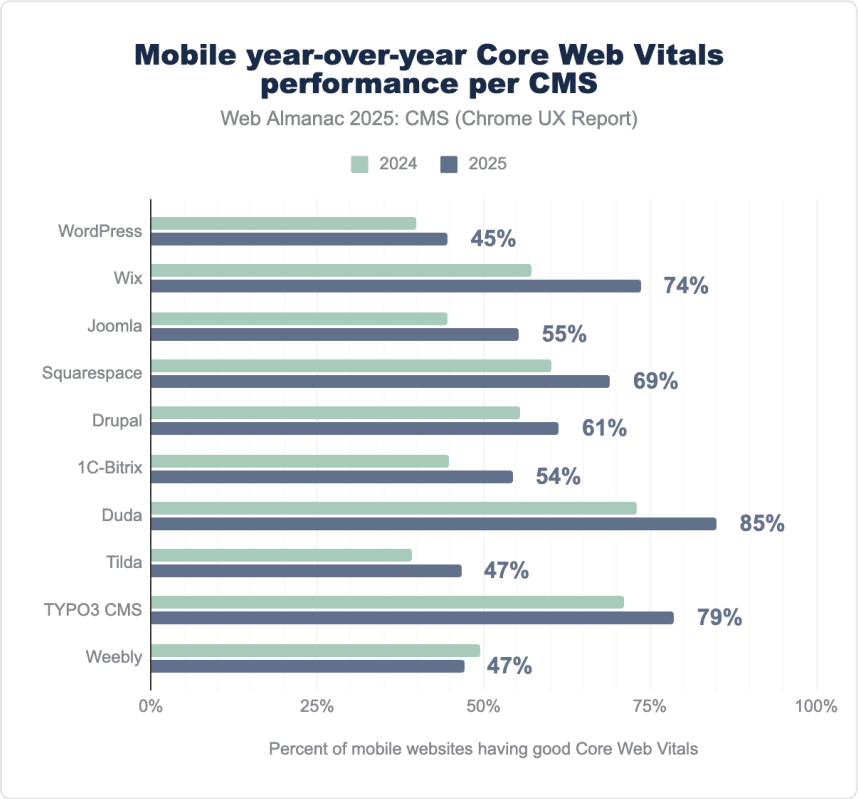


図12.8. CMS別モバイルのCore Web Vitalsパフォーマンスの前年比。

2024年から2025年にかけて、ほとんどのCMSプラットフォームがCore Web Vitalsの全体的なパフォーマンスを向上させましたが、その改善幅はさまざまです。より厳密に管理された環境を持つプラットフォームが最大の伸びを示しており、Wix（前年比約+14%）とDuda（+11%）が先頭に立ち、Squarespace（+8%）とJoomla（+7%）が続きます。1C-Bitrix、Tilda、TYPO3などのいくつかの小規模プラットフォームは約5%改善しました。

より拡張性の高いプラットフォームは改善幅が小さかったです。WordPressとDrupalはそれぞれ前年比約4%の伸びを示した一方、Weeblyはわずかに低下しました（約-1%）。これらのパターンは、カスタマイズと実装の多様性をより多く許可するエコシステムで改善を均等に伝えることがいかに難しいかを浮き彫りにしています。

Largest Contentful Paint（LCP）

Largest Contentful Paintはページのメインコンテンツがどれだけ早く表示されるかを測定し、体感的な読み込み速度の最も重要な指標の一つです。

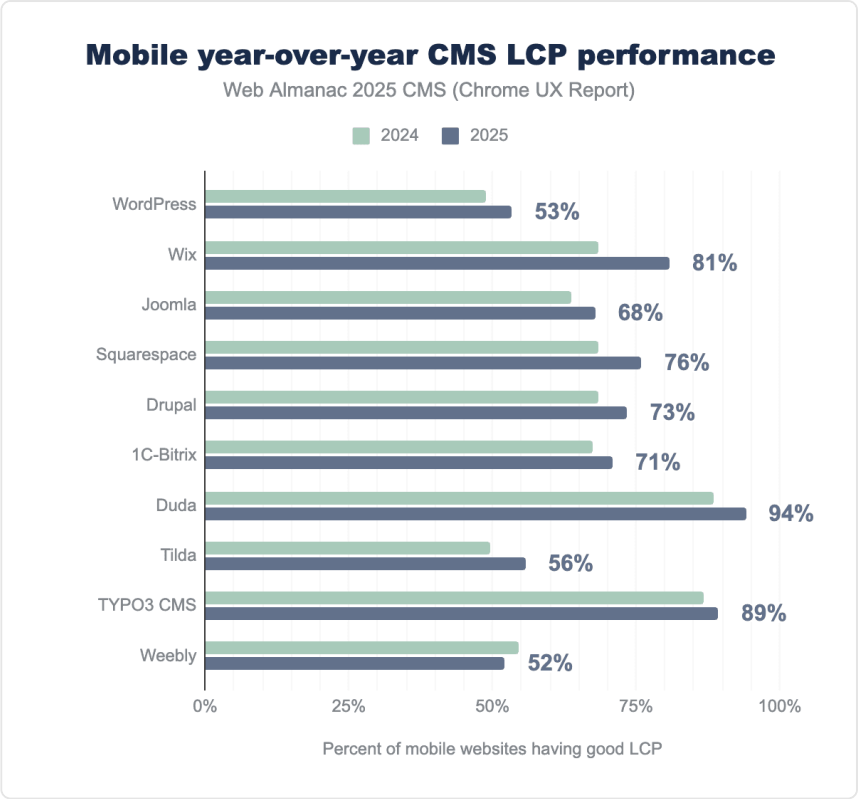


図12.9. CMS LCPパフォーマンスのモバイル前年比。

2025年には、54%のサイトが「良い」LCPスコアを達成し、継続的な進歩と同時に残された大きな変動を反映しています。

ほとんどのCMSプラットフォームは前年比でLCPが改善しています。Wixが約10%の伸びでリードし、Squarespace（+7%）とDuda（+5%）が続きます。WordPress、Joomla、Drupalはそれぞれ約4%改善し、Weeblyはわずかに低下しました（約-1%）。これらの差異は、アセット読み込み、画像最適化、デフォルト設定に関するプラットフォームレベルの選択と一致しています。

Cumulative Layout Shift（CLS）

Cumulative Layout Shiftは、予期しないレイアウトの動きを測定することで、ページが読み込まれる間どれだけ視覚的に安定しているかを捉えます。

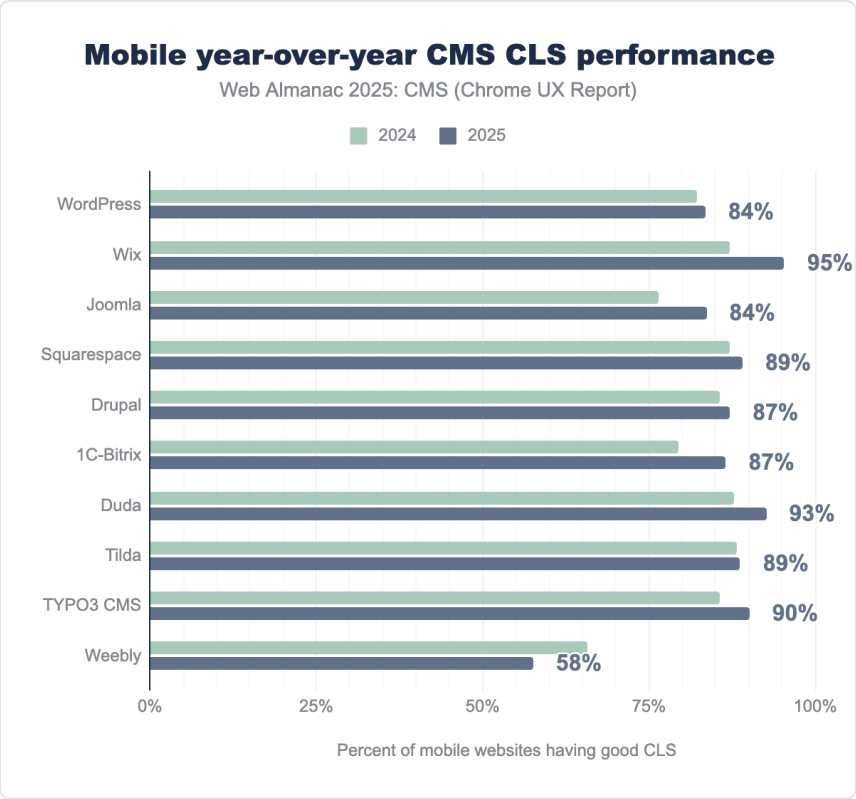


図12.10. CMS CLSパフォーマンスのモバイル前年比。

CLSはプラットフォーム間で最も不均一な指標の一つであり、遅延読み込みコンテンツ、埋め込み、動的レイアウトの管理における課題を反映しています。

Wixは再び最も強い前年比の改善を示し（約+8%）、Duda（約+4%）、Joomlaと1C-Bitrix（それぞれ約+3%）が続きます。他のプラットフォームはほとんど変化がなく、Weeblyは顕著な低下を経験しました（約-8%）。全体的に、CLSの結果はプラットフォームの選択だけでなく、実装の規律に大きく依存しているようです。

Interaction to Next Paint（INP）

Interaction to Next Paintは、最初の読み込み時だけでなく、すべてのユーザーインタラクションにおいてページがどれだけ応答性が高いかを測定します。

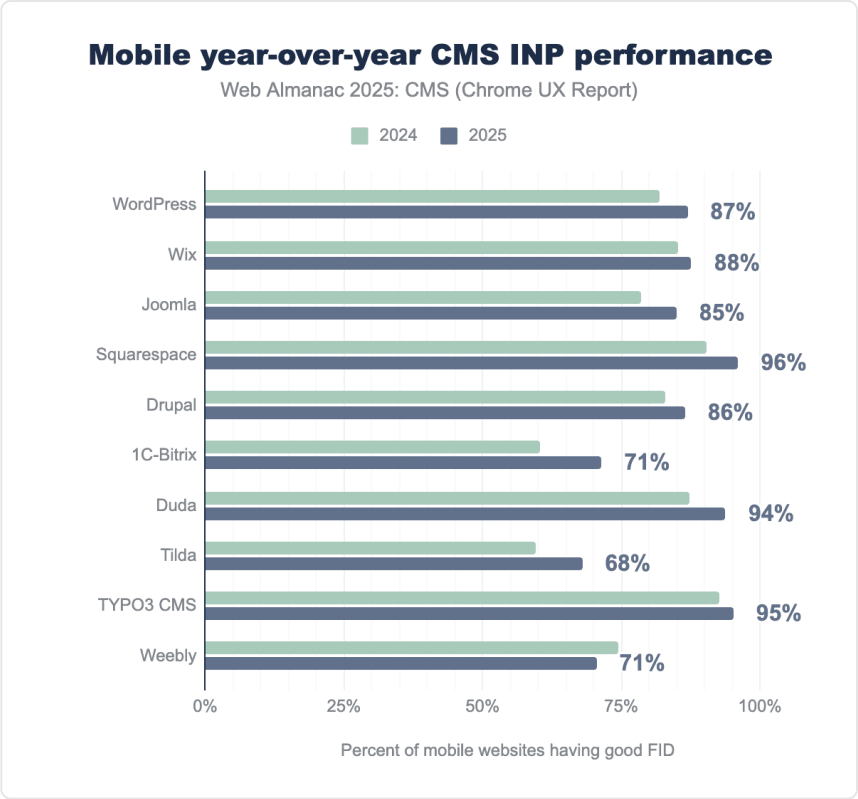


図12.11. CMS INPパフォーマンスのモバイル前年比。

LCPとCLSと比べると、INPの改善は控えめで、JavaScript、長いタスク、サードパーティのスクリプトを管理することがいかに難しいかを強調しています。

2025年には、1C-BitrixがINPの改善でリードし（約+10%）、SquarespaceとDuda（約+6%）、JoomlaとTilda（約+5%）、Drupal（+3%）が続きます。Weeblyは再び低下しました（-3%）。全体的に、どのCMSもスケールで一貫して優れたINPを提供しておらず、インタラクションの遅延が共通の問題として残っていることを示唆しています。

Lighthouseの品質指標

Lighthouseは、パフォーマンス、アクセシビリティ、SEO、ベストプラクティスにわたるサイト品質についての補完的なラボベースの視点を提供します。Lighthouseのスコアは実際のユーザーエクスペリエンスを直接反映するものではありませんが、一貫したテスト条件下での典型的な実装を比較するのに役立ちます。

パフォーマンス

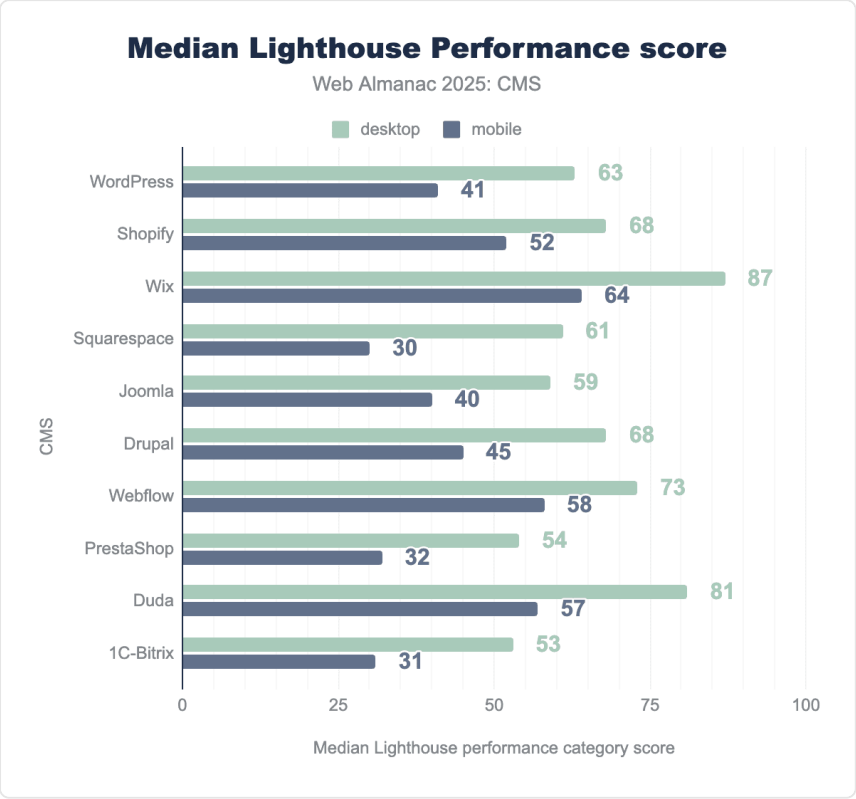


図12.12. Lighthouseパフォーマンススコアの中央値。

Lighthouseパフォーマンススコアの中央値は2024年から2025年にかけてデスクトップとモバイルの両方で改善され、デスクトップが一貫して高いスコアを示しています。デスクトップでは、Wix（87）とDuda（81）が先頭に立ち、Webflow（73）が続きます。WordPressは中央値スコア63を記録しており、複数のプラットフォームが近い値を示しています。

モバイルでは、スコアは全体的に低くなっています。Wixが64でリードし、Webflow（58）、Duda（57）、Shopify（52）が続きます。WordPress（41）とJoomla（40）はこのグループよりも低く、PrestaShopと1C-Bitrixが最低スコアを記録しています。前年比では、Wixがモバイルで最大の伸びを示し（55から64）、他のほとんどのプラットフォームはわずかな変化にとどまるか安定しています。これらのラボスコアは実際のユーザーエクスペリエンスの代替ではなく、コンテキストとして読むのが最善です。

SEO

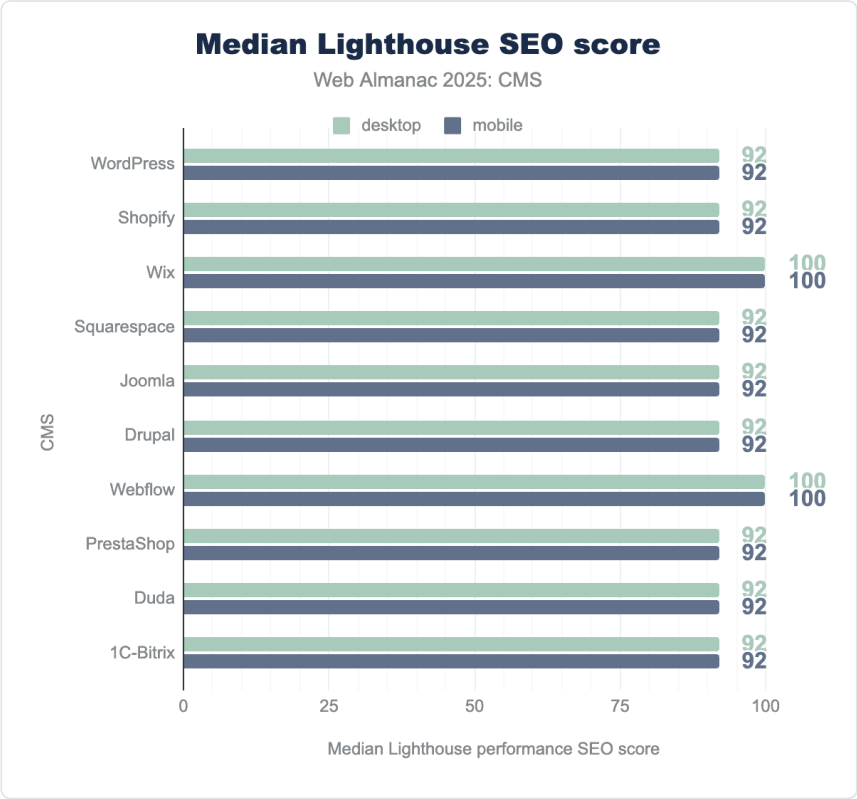


図12.13. LighthouseのSEOスコアの中央値。

2025年には、LighthouseのSEOスコアはCMSプラットフォーム全体で高いままで、ほとんどがモバイルとデスクトップの両方で92から100の間に集中しています。WebflowとWixが完璧なスコアを達成し、WordPress、Duda、Joomlaは約92で留まっています。小さな前年比の変化は、基本的なSEOのベストプラクティスが現代のCMSプラットフォームに広く組み込まれるようになったことを示唆しています。

アクセシビリティ

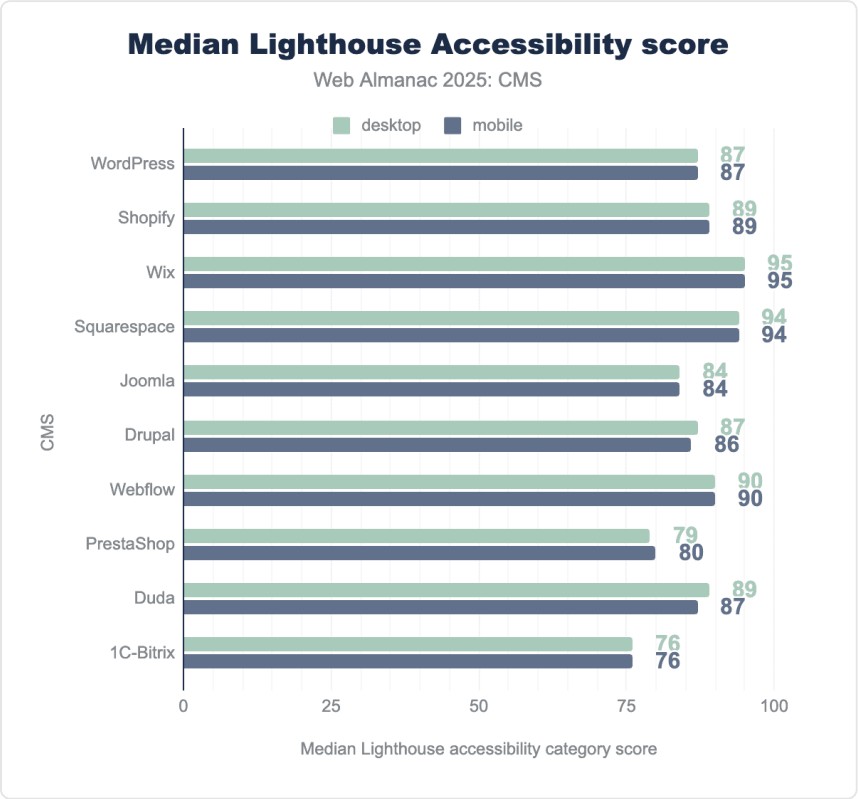


図12.14. Lighthouseアクセシビリティスコアの中央値。

アクセシビリティスコアはSEOよりも変動しますが、前年比ではまだ限られた変化しか示していません。2025年には、中央値スコアは76から95の範囲で、Wix（95）とSquarespace（94）がリードしています。WordPressとJoomlaは安定したままで、1C-Bitrixは76で後れを取っています。全体的に、改善は段階的で、着実だが変革的ではない進歩を示しています。

ベストプラクティス

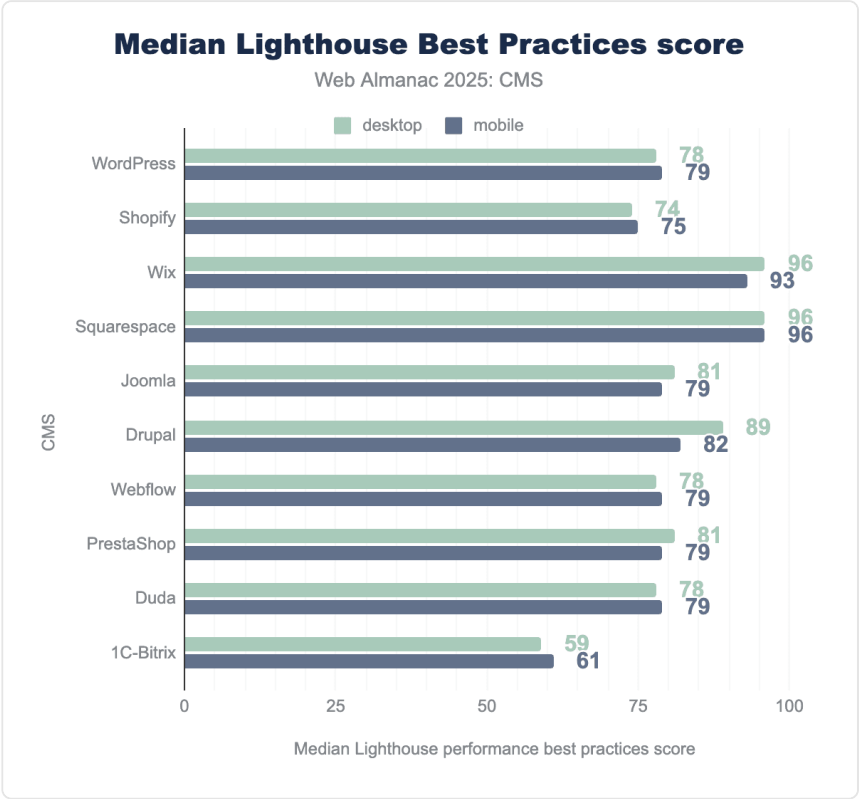


図12.15. Lighthouseベストプラクティススコアの中央値。

ベストプラクティスのスコアはより明確な差異を示しています。モバイルでは、Squarespace（96）とWix（93）がリードし、DrupalとPrestaShop（ともに82）が続きます。WordPress、Duda、Joomlaは約79の周りに集まり、1C-Bitrixが61で最低ランクです。デスクトップの結果も同様のパターンをたどっています。

前年比では、一部のプラットフォームが著しく改善しています。特にWix（79から93）とDrupal（79から82）が改善し、他のプラットフォームはほとんど動きがありません。これらの差異は、現代のAPI使用、セキュリティ設定、エラーハンドリングなどの分野での継続的なギャップを示しています。

ページ重量とリソース構成

「ページ重量」とは、ブラウザがページをレンダリングするためにダウンロードする必要があるすべてのリソースの合計サイズです。過去10年間、ページ重量は着実に増加しています。

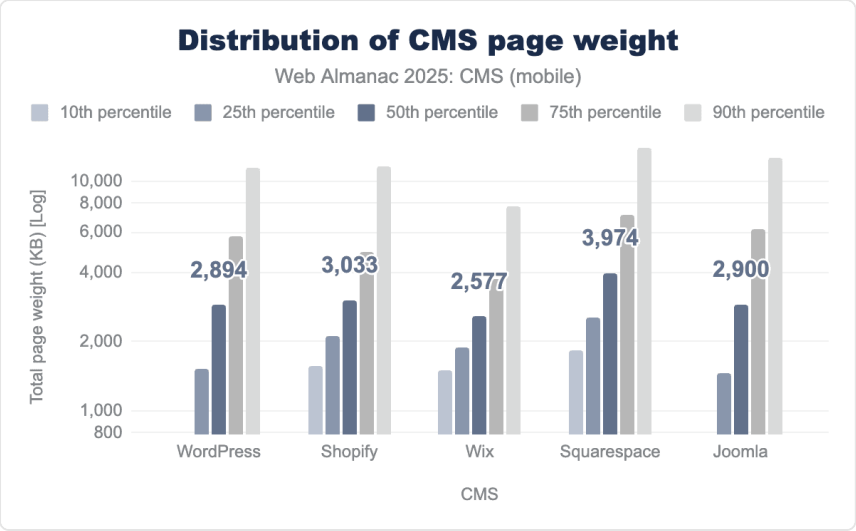


図12.16. CMSページ重量の分布。

2025年には、平均ページはデスクトップで約2.67 MB、モバイルで2.28 MBです。どちらの数値も一般的に推奨される1〜1.5 MBの範囲を超えており、ウェブ全体での継続的な肥大化を反映しています。

この成長の大部分は画像とJavaScriptから来ており、HTMLは合計転送サイズの比較的小さな部分のままです。パフォーマンスの問題への認識が高まっているにもかかわらず、ページ重量は増え続けています。3秒以上かかるページは直帰率が大幅に高くなる傾向があり、追加の遅延はコンバージョン率の低下と強く結びついています。低速または従量制接続のモバイルユーザーにとって、重いページはアクセスの実際の障壁になる可能性があります。

したがって、ページ重量は単なる技術的な詳細以上のものです。ユーザーの認識、アクセシビリティ、ビジネスの結果を形作ります。ページ重量を一流の制約として扱う組織は一般的にパフォーマンスがより予測可能になり、そうでない組織はしばしば遅い体験と関与の低下に苦労します。

CMSによるページ重量の合計

ページ重量の合計はCMSプラットフォーム間で大きく異なり、デフォルトテーマ、アセットパイプライン、プラグインエコシステム、ホスティングモデルに影響されます。すべてのプラットフォームでページ重量が増加した一方で、CMSの選択は依然として中央値サイズとページ重量の分布の両方に影響します。

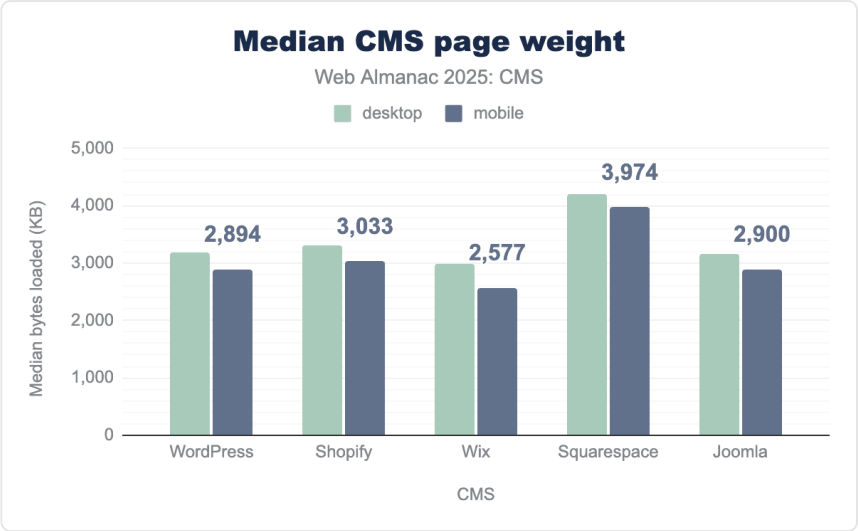


図12.17. CMSページ重量の中央値。

より制御された環境にあるプラットフォームはより狭い分布を示す傾向があります。より拡張性の高いCMSは、ページ重量がコアによってではなくテーマ、プラグイン、ページビルダー、サードパーティツールの累積的な影響によって促進されるため、より広い変動を示します。その結果、同じCMS上の2つのサイトで合計転送サイズが大きく異なる可能性があります。

この変動性は以前のパフォーマンスパターンを反映しており、ページ重量とより広いパフォーマンス結果の間の関連性を強化しています。

リソース構成：画像

画像はすべてのCMSプラットフォームでページ重量の最大の要因であり続けています。最新のフォーマットとレスポンシブ画像技術がより一般的になっていますが、その利点はしばしば画像数の増加と大きなビジュアルアセットによって相殺されます。特にテンプレートの多いサイトやビジュアルが豊富なサイトではその傾向が顕著です。

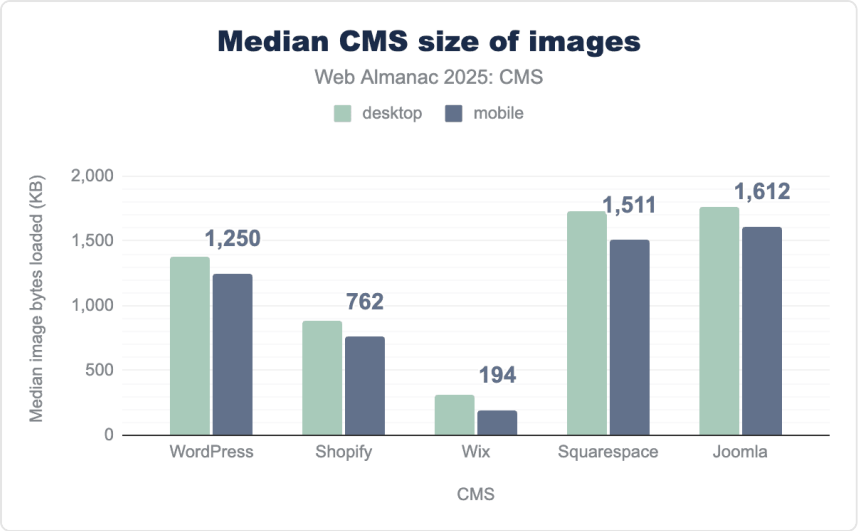


図12.18. CMS画像サイズの中央値。

画像処理のデフォルトが強いCMSプラットフォームは、一般的により小さく一貫した画像ペイロードを生成します。より柔軟なシステムでは、画像の最適化はテーマの選択、プラグインの設定、編集習慣に大きく依存しており、より大きな変動につながります。

画像ペイロードの継続的な成長は、一部のプラットフォームが読み込み指標を改善しながらも全体的にはより重いページを提供できる理由を説明するのに役立ちます。

リソース構成：JavaScript

JavaScriptはページ重量で最も急成長している部分であり、ランタイムパフォーマンスの最も重要な要因の一つです。JavaScriptペイロードは、コアCMSロジック、テーマ、ページビルダー、アナリティクス、広告、その他のサードパーティサービスの組み合わせたコストを反映しています。

ページビルダーとコンポーネントベースのシステムは、クライアントサイドのレンダリングとインタラクションレイヤーを追加することでJavaScriptの使用を増やすことが多いです。新しいアプローチは不要なスクリプトをより少なく読み込もうとしていますが、JavaScriptは前述のインタラクション遅延と応答性の問題の主要な要因であり続けています。

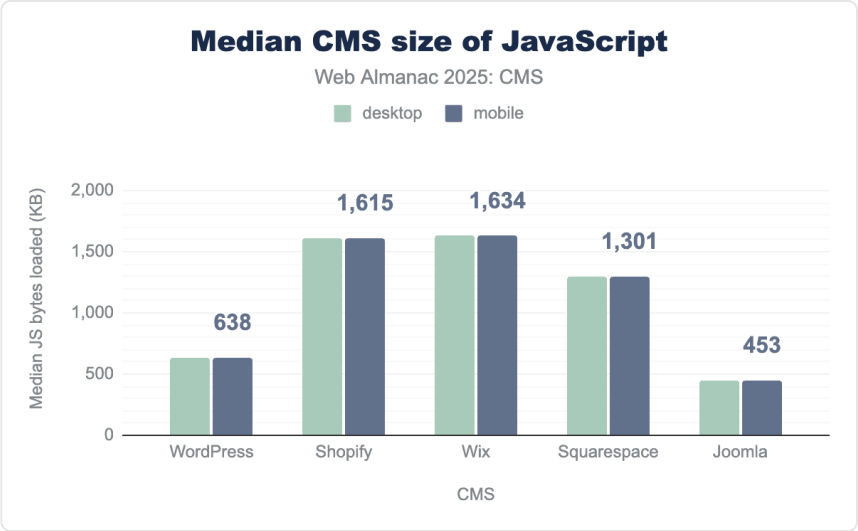


図12.19. CMSのJavaScriptサイズの中央値。

全体的なページ重量と同様に、JavaScriptのコストは拡張性の高いCMSエコシステム内で大きく異なります。実装の選択はプラットフォームの選択だけよりも重要です。

CSSとHTML

CSSとHTMLは画像やJavaScriptよりも合計ページ重量に占める割合は小さいですが、それでもレンダリングに影響します。大きなグローバルスタイルシート、重複ルール、スコープのないスタイルはレンダリングを遅延させ、ブロッキング時間を増やす可能性があります。

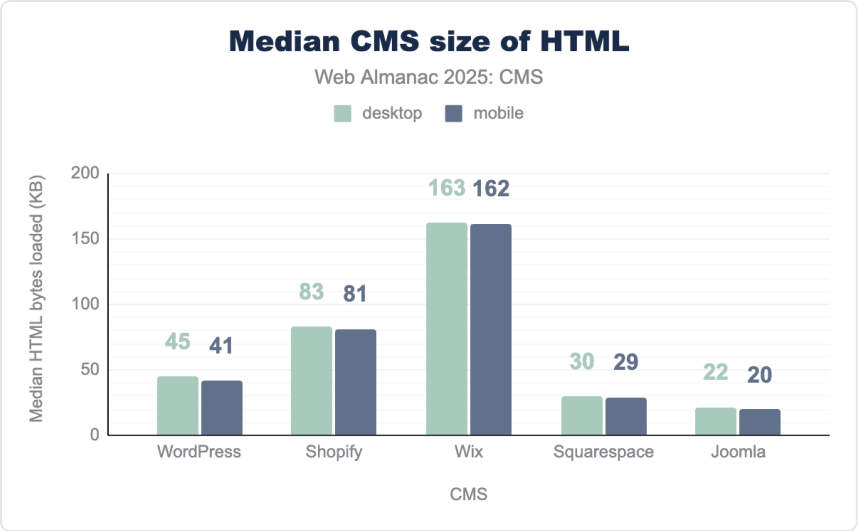


図12.20. CMSのHTMLサイズの中央値。

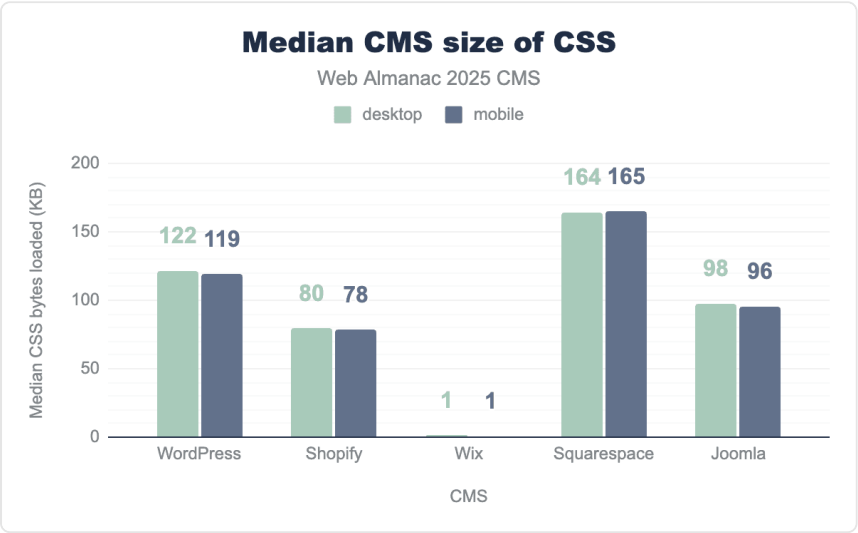


図12.21. CMSのCSSサイズの中央値。

ブロックベースおよびコンポーネント駆動のシステムは、CSSを動的に生成することが増えています。キャッシュと再利用が適切に行われる場合、これにより未使用のスタイルを削減できます。適切に処理されない場合、複雑さとオーバーヘッドが増加します。これらのトレードオフは、この章で先に探求した幅広いアーキテクチャのテーマを反映しています。

ページ重量、パフォーマンス、分散

ページ重量とパフォーマンスの関係は厳密に線形ではありませんが、強い相関があります。重いページはCore Web Vitalsの閾値を逃す可能性が高く、特にネットワークとCPUの制限が非効率性を増大させるモバイルデバイスでそれが顕著です。

CMS全体において、ページ重量は単一の障害点よりも複合的な要因として機能します。大きな画像ペイロード、重いJavaScript、グローバルスコープのアセットが時間とともに蓄積し、遅い読み込み、視覚的な不安定さ、遅いインタラクションの可能性を高めます。カスタマイズを制限するプラットフォームは一貫したデフォルトを通じてこのリスクを軽減する傾向があり、拡張性の高いシステムはサイトオーナーと実装者により多くの責任を課します。

全体的に、ページ重量はこの章の核心的なアイデアを強化しています：CMSプラットフォームはリソースがどのように組み立てられ提供されるかに影響することで、パフォーマンスを間接的に形成します。ページがより複雑になるにつれ、それに含まれるものを管理すること、そしてどのように提供されるかを管理することは、信頼性の高いパフォーマンスを維持するために重要です。

新興ウェブAPI

CMSプラットフォームが成熟するにつれ、ブラウザの機能がパフォーマンスとユーザーエクスペリエンスの形成においてより大きな役割を果たしています。新しいウェブAPIはフレームワークやCMSを横断して機能する改善をますます提供しており、最適化作業の一部をサーバーからブラウザ自体にシフトしています。これらのAPIは根本的なコストを除去するわけではありませんが、慎重に使用すると知覚される遅延を減らして応答性を向上させることができます。

このセクションでは、CMSで構動されるサイト全体のナビゲーション速度、インタラクションの応答性、知覚されるパフォーマンスに影響を与え始めているブラウザレベルの機能を紹介します。

Speculation Rules API

ナビゲーションの遅延は、コンテンツが多い複数ページのサイト（多くのCMSが動かすタイプ）での最も目立つ遅さの原因の一つであり続けています。Speculation Rules APIは、ブラウザがプリフェッチまたはプリレンダリングを通じて、起こりそうなナビゲーションを予測してページを事前に準備できるようにすることでこれに対処します。

従来のプリロードとは異なり、投機ルールにより開発者はどのナビゲーションが起こりそうか、またどのような条件下でブラウザが行動すべきかを宣言できます。初期の使用では、一

般的な閲覧パスのFirst Contentful PaintとLargest Contentful Paintの低下、よりスムーズなページ遷移など、ナビゲーションパフォーマンスの測定可能な改善が示されています。

CMSにとってのSpeculation Rules APIの主な価値は、バックエンドのレンダリングロジックを変更せずに知覚されるナビゲーション速度を向上させることです。投機的な読み込みはブラウザで行われるため、特に予測可能なナビゲーションフローを持つサイトでデータベースの遅延、プラグインのオーバーヘッド、またはページビルダーの複雑さの影響を軽減できます。このAPIは現在Chromiumベースのブラウザでサポートされており、他の場所では安全に無視されるため、プログレッシブエンハンスメントに適しています。

Long Animation Frames API（LoAF）

多くのCMSプラットフォームで読み込みパフォーマンスが改善されている一方、インタラクションの応答性は依然として一貫した問題点です。Long Animation Frames APIは、ビジュアル更新とユーザーフィードバックをブロックする遅延したアニメーションフレームを追跡することで、以前の長いタスク測定を基に構築されています。これはInteraction to Next Paint（INP）を理解することに直接つながっています。

LoAFは個々のJavaScriptタスクからフル フレームの遅延に注目をシフトさせ、何が本当に応答性に影響するかを特定しやすくします。どのスクリプト、レイアウト操作、またはレンダリングステップが最も頻繁に遅いフレームを引き起こすかを強調することで、LoAFはチームが初期ページ読み込みテストで表示されない可能性のある問題を明らかにするのに役立ちます。

これは、遅さが単一のブロッキング操作よりも累積的な複雑さから生じることが多いCMS駆動サイトで特に重要です。ページビルダー、アナリティクス、広告、サードパーティのツールは、時間とともに応答性を劣化させる方法で相互作用する可能性があります。LoAFを使用することでチームはリアルユーザーモニタリングでこれらのパターンを確認し、読み込み時だけでなくセッション全体でインタラクションがどのように機能するかを理解できます。

View Transitions API

View Transitions APIは、ブラウザがコンテンツの変更をアニメートできるようにすることで、ページ状態間のよりスムーズなビジュアルトランジションを可能にします。これは生の速度よりも視覚的な連続性に関するものですが、特にユーザーが頻繁にナビゲートするコンテンツの多いサイトで知覚されるパフォーマンスを大幅に改善できます。

CMSプラットフォームにとってビュートランジションが重要なのは、従来のマルチページサイトとシングルページアプリケーションの間のエクスペリエンスのギャップを縮小するためです。投機的な読み込みなどの技術と組み合わせることで、複雑なSPAアーキテクチャを採用せずに、サーバーレンダリングされたCMS駆動サイトをよりスムーズに感じさせることが

できます。ブラウザ間でのサポートはまだ不均一で使用は初期段階ですが、初期の採用は知覚されるパフォーマンス改善への関心の高まりを示唆しています。

Priority HintsとScheduling API

ページがより複雑になるにつれ、リソースの優先順位付けがパフォーマンスでより大きな役割を果たすようになります。Priority Hintsにより開発者はどのリソース（画像、フォント、スクリプトなど）がより重要かを示すことができ、ブラウザが制約された条件下で読み込み順序を調整できるようにします。適切に使用すると、合計ページ重量を削減することなく主要なレンダリングのマイルストーンを改善できます。

同様に、新しいスケジューリングAPIはメインスレッドの作業がいつ行われるかについてより多くの制御を提供し、重要でないタスクをユーザーに見えるインタラクションのために延期しやすくします。多くの機能レイヤーを持つCMS駆動のページでは、これらのツールは主要なインタラクションの瞬間における競合を軽減するのに役立ちます。

CMSプラットフォームへの示唆

これらのAPIを合わせると、ウェブパフォーマンスが時間とともにどのように改善されるかというより広いシフトを示しています。バックエンドの変更やCMS固有の最適化にのみ依存するのではなく、プラットフォームはユーザーの行動とデバイスの能力に適応するブラウザインテリジェンスにますます頼ることができます。これらのAPIは大きなペイロードや重い実行のコストを消去するわけではありません。

CMSエコシステムにとって、これは馴染みのあるパターンを強化しています：ブラウザAPIは拡張性によって導入される変動性の一部をスムーズにできますが、柔軟性と予測可能性の間のコアトレードオフを排除することはできません。その影響は慎重な設定、ユーザー行動についての現実的な仮定、既存アーキテクチャへの適合に依存します。

ブラウザが進化し続けるにつれ、CMSプラットフォームはコアデザインを再構築せずにナビゲーションをスムーズにし、応答性を向上させ、知覚される遅延を減らすためにこれらのAPIを段階的に採用する可能性が高いです。その意味で、新興ウェブAPIは破壊的な力というよりも、既存のパフォーマンス戦略の乗数として機能します。

CMSにおける人工知能

AI機能はCMSプラットフォーム全体でより一般的になっていますが、現在はコアデリバリーパフォーマンスよりもコンテンツワークフローに多く影響を与えています。ほとんどの場合、AIはコンテンツがレンダリングまたはユーザーに提供される方法ではなく、コンテンツがどのように作成、整理、充実されるかをサポートするために使用されています。

エコシステム全体で、AIの採用は不均一でほとんどがオプションです。AIは通常、CMSコアへの根本的な変更としてではなく、プラグイン、拡張機能、または外部サービスを通じた支援的なレイヤーとして現れます。一般的な使用法には、コンテンツの草稿作成と編集、要約、翻訳、画像生成、タグ付け、SEOメタデータの提案が含まれます。

ほとんどのAIワークフローはオーサリング中または非同期バックエンドで行われるため、ページ重量、リソース構成、Core Web Vitalsに大きな影響を与えません。AIがパーソナライゼーション、レコメンデーション、または動的コンテンツのためにランタイムで使用される場合、その影響は実装に応じて追加のJavaScriptやネットワークリクエストを通じて間接的に現れる可能性が高いです。

AIが他のCMSトレンドと交差する一つの領域は編集のスケールです。生産および更新されたコンテンツのアップロードコストと他のチャネルへの配信時間を下げることで、AIはより多くのページ、よりリッチなメディア、よりダイナミックなエクスペリエンスにつながる可能性があります。これらの変化は、ページ重量、クライアントサイドの実行、キャッシュに関する既存のパフォーマンスの課題を激化させる可能性があり、この章の他の場所で見られるパターンを反映しています。

今のところ、CMSプラットフォームにおけるAIはパフォーマンスオプティマイザーよりもワークフローアクセラレーターとして理解するのが最善です。ユーザーエクスペリエンスへの影響は、読み込み速度や応答性の直接的で測定可能な向上ではなく、編集の慣行、ガバナンス、実装の規律を通じて媒介されます。AIがランタイムシステムに近づくにつれ、そのパフォーマンスへの影響は大きくなる可能性がありますが、現在の証拠はその主な影響が依然として運用面にあることを示唆しています。

LLM検索時代のCMS

大規模言語モデル（LLM）ベースの検索と回答エンジンの台頭により、CMS生成コンテンツが評価・使用される方法が変化しています。ページをランク付けすることに焦点を当てた従来の検索エンジンとは異なり、LLMシステムはコンテンツを抽出、要約、再結合します。これにより、構造、明確さ、新鮮さ、機械可読シグナルの重要性が高まっています。

HTTP Archiveの観点からは、これらの変化は構造化データのより多くの使用、より明確なコンテンツ階層、より一貫したセマンティックマークアップとして現れています。適切に構造化されたHTML、論理的な見出し順序、リッチなメタデータを促進するCMSプラットフォームは、コンテンツがLLMシステムによって正確に解釈・再利用されるためにより良い位置にあります。

インデックス動作も変化しています。LLMベースのツールは定期的に更新され、明確に帰属され、抽出しやすいコンテンツを好む傾向があります。迅速な更新とプログラマティックなメタデータ生成をサポートするCMSは間接的な利点を得る可能性があります。一方で、クラ

イアントサイドのレンダリング、大きなJavaScriptバンドル、または遅延したコンテンツのハイドレーションに大きく依存するページは、抽出が遅くなるか不完全になる可能性があるため、可視性が低下する場合があります。

これらのダイナミクスは、前述のパフォーマンスパターンと密接に一致しています。JavaScriptの肥大化、遅いLCP、または低いINPに既に苦しんでいるCMSは、LLMシステムが効率性と明確さをますます報酬とするにつれて、複合的な発見可能性の問題に直面する可能性があります。LLM検索は従来のインデックスを置き換えるわけではありませんが、柔軟性、パフォーマンス、コンテンツ構造の間の既存のトレードオフを増幅させます。

構造化データの進化

構造化データは、CMSプラットフォーム、検索エンジン、AI駆動のコンシューマー間の重要なレイヤーになっています。2025年までに、それはもはやリッチスニペットや強化された検索結果のためのツールだけではありません。コンテンツが自動化されたシステム全体でどのように解釈、要約、表示されるかを管理する機械可読コンテキストとしてますます機能しています。

HTTP Archiveのデータは、CMS全体での構造化データの使用の着実な成長を示していますが、実装の品質は大きく異なります。ネイティブスキーマサポートを持つプラットフォームは、より一貫して予測可能なマークアップを生成する傾向があります。プラグインまたはカスタム実装に大きく依存するプラットフォームはより大きな変動を示しており、柔軟性がしばしば一貫性のコストを伴うというより広いパフォーマンスパターンを反映しています。

構造化データはパフォーマンスとページ重量とも相互作用します。適切に実装されていないスキーマは、具体的なメリットなしにHTMLのサイズを肥大化させ、DOMの複雑さを増やす可能性があります。適切にスコープされた構造化データは、対照的に、コンテンツをより解釈可能にしながら最小限のオーバーヘッドを追加します。CMSがスキーマ生成をより自動化するにつれ、しばしばAI支援ツールを通じて、リスクは構造化データがないことから多すぎることにシフトし、冗長または不要なマークアップが結果を改善せずに重量を追加します。

進化する検索と発見システムの世界において、構造化データは安定化シグナルとして機能します。スキーマを検証し、冗長性を制限し、アドホックなプラグインに散在するのではなくコアテンプレートに構造化データを統合するCMSプラットフォームは、より耐久性のある機械フレンドリーなコンテンツを生成する可能性が高いです。ますます、構造化データの成熟度は、SEOだけでなく、自動化された消費が成長するにつれてコンテンツの長期的な耐久性のための差別化要因になっています。

結論

2025年のCMSランドスケープは、成熟しつつも二極化が進むウェブを反映しています。HTTP Archiveのデータは、CMSプラットフォームが今やウェブサイトの大多数を支えている一方で、非CMSサイトはウェブのシェアとして縮小し続けていることを示しています。同時に、採用パターン、パフォーマンス結果、リソース使用量はプラットフォームの選択、ホスティングモデル、実装の規律によって大きく異なります。

市場シェアデータはWordPressがCMS主導サイトの60%以上を動かす支配的なCMSであり続けることを確認しています。Shopifyなどのプラットフォームは特定のニッチ、特にEコマースで急速に成長し続けている一方で、多くの他のホスト型ビルダーは減速の兆候を示しています。ランクベースの分析はこれらのトレンドが不均一であることを明らかにしています：オープンソースCMSは高トラフィックサイトで過剰に代表され続け、SaaSビルダーはより小さなプロパティの長いテールで支配しています。これはCMSの選択が単なる一般的な人気ではなく、組織のニーズと制約によってますます促進されていることを示唆しています。

パフォーマンスデータはこの分割を強化しています。垂直統合型プラットフォームは特にモバイルで、より一貫したCore Web Vitalsを提供する傾向があります。セルフホスト型CMSはテーマ、プラグイン、サードパーティ統合に影響されてより大きな変動を示します。ページ重量とリソース構成データはさらに、実装の決定がしばしばプラットフォームのデフォルトよりも重要であることを示しています。単一のCMS内でも、サイトは高度に最適化されたものから著しく肥大化したものまで様々で、パフォーマンスは組み込みの保証ではなく継続的なプラクティスであることを強調しています。

Core Web VitalsとLighthouseの結果は、繰り返すテーマを浮き彫りにしています：CMSの選択は主に提供する制約、デフォルト、ツールを通じてパフォーマンスに影響します。より制御された環境を持つプラットフォームはより安定した結果を生み出す傾向があります。より拡張性の高いシステムは逆に、優れた実装と低品質な実装の両方を可能にします。

これらのパターンは単純な勝者と敗者を示すものではありません。代わりに、柔軟性、制御、予測可能性の間のトレードオフを反映しています。CMSエコシステムが多様化するにつれ、結果はサイト底部のロゴよりも、各プラットフォームのツールと制約がサイトの複雑さ、チームの能力、長期的なメンテナンス目標にどれだけ合っているかにより依存するようになっています。

現代のウェブAPI、AI支援ワークフロー、LLMベースのコンテンツ消費の重要性の高まりはさらなる圧力を加えています。構造化データ、セマンティックマークアップ、効率的なコンテンツ配信はもはや「あれば良い」最適化ではありません；それらは従来の検索と新しいAIインターフェースの両方でコンテンツが発見可能、解釈可能、パフォーマンスが高いかどうかをますます決定しています。一貫した構造を促進し、不必要な複雑さを制限し、現代のブラウザ機能を採用するCMSプラットフォームは適応するためにより良い位置にあります。

全体的に、2025年のCMSエコシステムは純粋な成長ではなくトレードオフによって定義されています。柔軟性はしばしばパフォーマンスの変動を増加させます。使いやすさは重いページを生み出す可能性があります。自動化は新しいガバナンスと品質の課題を導入します。CMSプラットフォームは進化し続けますが、実際の結果における支配的な要因は実装であり続けます。ウェブが速度、安定性、機械可読コンテンツにより重点を置くにつれ、CMSはコアインフラであり続けます。それは複雑さを隠すからではなく、その複雑さがスケールでどのように現れるかを決定するからです。

著者

Vahe Arabian

VaheSODP ahearabian https://www.stateofdigitalpublishing.com/

Vahe Arabianはデジタルパブリッシングの起業家、成長戦略家であり、State of Digital Publishing[453]とSODP Media[454]の創設者です。パブリッシャー、スタートアップ、メディア組織と協力して、SEO、オーディエンス開発、マネタイズ、製品戦略にわたる持続可能な成長を推進しています。デジタルメディアトレンド、AIとパブリッシング、オンラインジャーナリズムの進化について頻繁に講演・執筆しています。

453. *https://www.stateofdigitalpublishing.com/*
454. *https://www.sodpmedia.com/*

部 III 章 13
Eコマース

Amandeep Singh によって書かれた。
Barry Pollard によってレビュ。
Amandeep Singh による分析。
Barry Pollard 編集。
Sakae Kotaro によって翻訳された。

はじめに

Eコマースはウェブにおける特別なケースではなくなりました。今やEコマース こそがウェブです。2025年には、購買の旅は検索結果、ソーシャルフィード、ライブストリームから始まり、音声アシスタント、メッセージアプリ、スマートTVのようなリーンバック型のサービスへと続き、さらにはAIエージェントが買い物客の代わりに完結させることも増えています。Eコマースサイトは依然として物理的またはデジタル製品を販売するオンラインストアですが、今やプロダクトページ、決済、パフォーマンス、アクセシビリティ、信頼の交差点に位置しています。

オンラインストアを構築する際、いくつかの一般的なプラットフォームモデルがあります：

1. **Software-as-a-Service（SaaS）** プラットフォーム（例：Shopify）は、コードベースを管理してホスティングを抽象化することで、ストア運営に必要な技術的知識を最小化します。

2. **Platform-as-a-Service（PaaS）** ソリューション（例：Adobe Commerce / Magento）は、完全なコードアクセスを許可しながら最適化されたテクノロジースタックとホスティング環境を提供します。
3. **セルフホスト型**プラットフォーム（例：WooCommerce、OpenCart）は、マーチャントまたはその代理店が管理するインフラストラクチャ上で動作します。
4. **ヘッドレス / APIファースト**プラットフォーム（例：Commercetools、Medusa）は、コマースバックエンドをサービスとして提供し、マーチャントがフロントエンドのエクスペリエンスとホスティングを所有します。
5. **エージェンティックコマース（エージェント対応コマース）** レイヤーはストアフロントの横（または上）に位置します：構造化された製品データ、在庫、ポリシー、アイデンティティ、決済フローがAPIと標準を通じて公開され、アシスタントとAIエージェントが明確なユーザーの同意とガードレールのもとで製品を安全に見つけて購入を実行できます。

プラットフォームは複数のカテゴリーに該当する場合があります。たとえば、一部のベンダーはSaaS、PaaS、セルフホスト型オプションを提供しており、多くのヘッドレスビルドは内部でSaaSバックエンドに依存しています。重要な変数は、誰がホスティングを管理するか、誰がランタイムとアップグレードパスを管理するか、フロントエンドとバックエンドを変更する自由がどれだけあるかです。

プラットフォームの検出

ウェブサイトが使用する技術を検出するためにWappalyzerというツールを使用しました。Eコマースプラットフォームやコンテンツマネジメントシステムや JavaScriptフレームワーク、アナリティクスなどを検出できます。

この分析では、以下のいずれかが検出された場合にサイトをEコマースと見なしました：

- 既知のEコマースプラットフォームの使用、または
- オンラインストアを強く示唆する技術の使用（例：拡張Eコマースアナリティクス）。

Eコマースサイトの認識における制限事項

私たちの方法論には精度に影響する制限事項があります。

- WappalyzerがEコマースプラットフォームまたは強力なEコマースシグナルを検出した場合にのみEコマースサイトを認識できます。

- 決済処理業者だけの検出（例：PayPal）はサイトをEコマースとして分類するのに十分ではありません。なぜなら多くの非ストアサイトも支払いを受け付けるからです。
- ストアがサブディレクトリでホストされている場合、見逃される可能性があります。サイトごと、クライアント（デスクトップとモバイル）ごとにホームページと他の1ページ（最大の内部リンク）をクロールします。
- ヘッドレス実装はHTML/JSの従来のフィンガープリントが消えることが多いため、プラットフォームの検出可能性を低下させます。
- 明らかなトレンドは、業界の変化だけでなく、検出の改善や後退によって影響を受ける場合があります。
- クロールの地理は重要です：サイトが位置に基づいてリダイレクトする場合、結果が異なる場合があります。
- 基礎となるサイトセットはChromeのフィールドデータエコシステムから抽出されており、Chromeユーザーが訪問するサイトに偏りがあります。

全体的な採用状況

図13.1. Eコマースサイトであるモバイルページの割合。

2025年のデータセットでは、分析した全デスクトップサイトの19.9%と全モバイルサイトの19.2%がEコマースサイトとして検出されました。

この数字は、Eコマースが単なる「一つのバーティカル」ではなく、オープンウェブで実際のユーザーが体験するものの大きな部分を占めていることを改めて示しています。

ランク別の採用状況

一般的に、最も人気のあるサイトはプロフェッショナルに設計され、大幅に最適化され、より多くの予算で支えられている可能性が高いです。

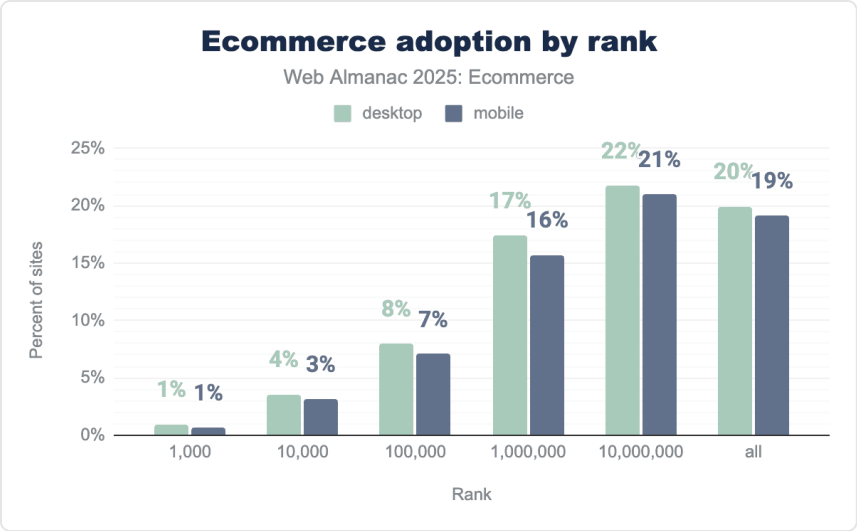


図13.2. ランク別のEコマース採用状況。

パターンは一貫しています：

- Eコマースであるサイトの割合は、人気の低いサイトを含めるほど増加します。
- ウェブの最上位（上位1,000）では、Eコマースは存在しますが稀です。
- ランクが下がるにつれて増加します。
- 上位1,000万に達する頃には、5サイトに1つ程度がオンラインストアです。

採用トレンド

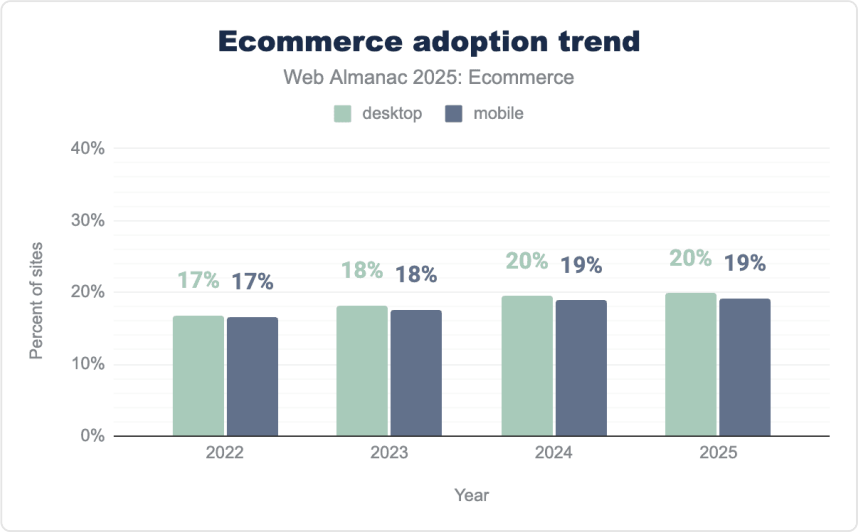


図13.3. 年別のEコマース採用状況。

時間の経過に伴うトレンドを見ると、前年比で緩やかな増加が見られます。

プラットフォームの市場シェア

デスクトップとモバイルの両方で、プラットフォームのランドスケープは上位に偏ったままです：少数のシステムが検出されたストアの大半を占め、ニッチおよび地域的なプラットフォームの長いテールが残りを埋めています。

以下の表は、Eコマース内で検出されたEコマースサイトのシェア（プラットフォームの市場シェア）と、データセット内のすべてのサイトで各プラットフォームが表示される頻度を示しています。

プラットフォーム	Eコマースサイトの%	全サイトの%
WooCommerce	35.4%	7.1%
Shopify	21.5%	4.3%
Squarespace Commerce	9.1%	1.8%
Wix eCommerce	7.8%	1.6%
PrestaShop	3.2%	0.6%
1C-Bitrix	2.2%	0.5%
Magento	2.1%	0.4%
OpenCart	1.1%	0.2%
Cafe24	1.0%	0.2%
BigCommerce	0.8%	0.2%

図13.4. デスクトップで最も人気のあるEコマースプラットフォーム。

プラットフォーム	Eコマースサイトの%	全サイトの%
WooCommerce	44.4%	7.0%
Shopify	25.3%	4.0%
Wix eCommerce	11.1%	1.8%
Squarespace Commerce	9.9%	1.6%
PrestaShop	3.7%	0.6%
1C-Bitrix	3.3%	0.5%
Magento	2.2%	0.3%
OpenCart	1.4%	0.2%
Tiendanube	1.0%	0.2%
Square Online	0.9%	0.1%

図13.5. モバイルで最も人気のあるEコマースプラットフォーム。

2024年以降のトレンド

ここ数年を俯瞰すると、話は混乱よりも緩やかな統合に関するものです。

- WooCommerceは最大のエコシステムであり続け、ほぼ横ばいを維持しています（2024年から2025年にかけてEコマースサイトの約36% → 36%）。
- Shopifyはシェアを拡大し続けています（約20% → 21%）。
- Wix eCommerceはトップ5で最も急成長しています（約7% → 8%）。
- PrestaShopはシェアが下降トレンドを続けています（約4% → 3%）。

言い換えると：デフォルトの選択肢がさらにデフォルトになっており、小規模なオープンソースのエコシステムは開発者エクスペリエンス、ホスティングのシンプルさ、デフォルトのパフォーマンスでより激しく競争しなければならなくなっています。

ティア別上位プラットフォーム

ティアが異なれば上位のプラットフォームも異なります。

順位	上位1,000	上位10,000	上位100,000	上位1,000,000	上位10,000,000	全体
1	Amazon Webstore	Amazon Webstore	Shopify	Shopify	WooCommerce	WooCommerce
2	Magento	Salesforce Commerce Cloud	Magento	WooCommerce	Shopify	Shopify
3	Pattern by Etsy	SAP Commerce Cloud	Salesforce Commerce Cloud	Magento	Squarespace Commerce	Squarespace Commerce
4		Magento	Amazon Webstore	PrestaShop	PrestaShop	Wix eCommerce
5		Shopify	WooCommerce	1C-Bitrix	Wix eCommerce	PrestaShop

図13.6. ランクティア別上位プラットフォーム（デスクトップ）。

順位	上位1,000	上位10,000	上位100,000	上位1,000,000	上位10,000,000	全体
1	Magento	Amazon Webstore	Shopify	Shopify	WooCommerce	WooCommerce
2	Amazon Webstore	Salesforce Commerce Cloud	Magento	WooCommerce	Shopify	Shopify
3	Pattern by Etsy	SAP Commerce Cloud	Salesforce Commerce Clous	Magento	Squarespace Commerce	Wix eCommerce
4		Magento	Amazon Webstore	PrestaShop	PrestaShop	Squarespace Commerce
5		Shopify	WooCommerce	1C-Bitrix	Wix eCommerce	PrestaShop

図13.7. ランクティア別上位プラットフォーム（モバイル）。

- 最上位のランクでは、エンタープライズおよびカスタムエコシステムがより多く登場します。
- より広いウェブでは、ロングテールの勝者（特にWooCommerce）が数量で支配しています。

地域別上位プラットフォーム

プラットフォームの優勢は、言語、地域の決済レール、エージェンシーエコシステム、ベンダーの歴史的なフットプリントにより地域によって異なります。

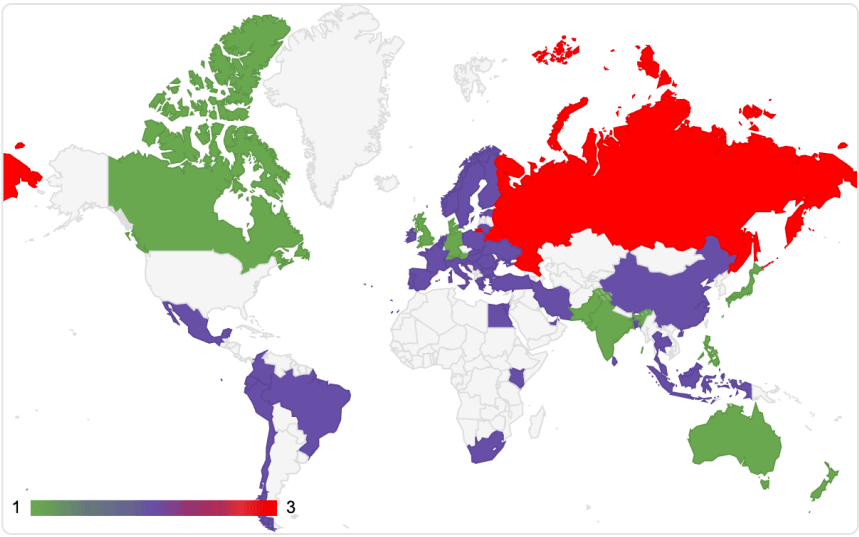


図13.8. 2025年の国別上位Eコマースプラットフォーム。

3つの主要プラットフォームがほとんどの国でトップを占めています：WooCommerce（バイオレット）、Shopify（グリーン）、1C-Bitrix（レッド）。

- デスクトップでは、WooCommerceは国別ビューの59の地域のうち42の地域で最も一般的なプラットフォームです（ALL集計を除く）。
- モバイルでは、WooCommerceはさらに多くの地域でリードしています：89のうち71（ALL集計を除く）。

意味のある地域的な例外もあります：

- 1C-Bitrixは東ヨーロッパと中央アジアの一部（例：ロシア連邦、ベラルーシ、カザフスタン、キルギスタン）でリードしています。
- Tiendanubeはアルゼンチンで際立っています。
- Shoptetはチェコで主要プラットフォームとして登場します。
- Cafe24は韓国でリードしています。
- Sallaはサウジアラビアで強い存在感を示しています。

Eコマースにおける**Core Web Vitals**

Eコマースサイトはパフォーマンスに特に敏感です。なぜなら1秒の追加ごとに影響が複合するからです：カテゴリーページが遅くなると商品閲覧数が減り、商品ページが遅くなるとカートへの追加が減り、チェックアウトフローが遅くなるとコンバージョンが減ります。

実際のユーザーエクスペリエンスを要約するためにCore Web Vitals（CWV）フィールド指標を使用します：

- **LCP（Largest Contentful Paint）：** *読み込み* パフォーマンスを測定します。メインコンテンツがどれだけ早く表示されるかを捉えます。Eコマースでは、ヒーロー画像、商品グリッド、レンダリングをブロックする重要なCSS/JSにマッピングされることが多いです。
- **INP（Interaction to Next Paint）：** *応答性* を測定します。ユーザーアクション（タップ/クリック）と次のビジュアル更新の間の遅延を捉えます。重いJavaScript、サードパーティタグ、メインスレッドの競合に敏感です。
- **CLS（Cumulative Layout Shift）：** *視覚的安定性* を測定します。ページの読み込み中にコンテンツがどれだけシフトするかを捉えます。遅れて読み込まれる商品画像、パーソナライゼーションウィジェット、プロモバナーが誤クリックを引き起こす可能性があるため、Eコマースで特に重要です。

CWVで「良い」とみなされるサイトは3つのすべての閾値を通過したものです。

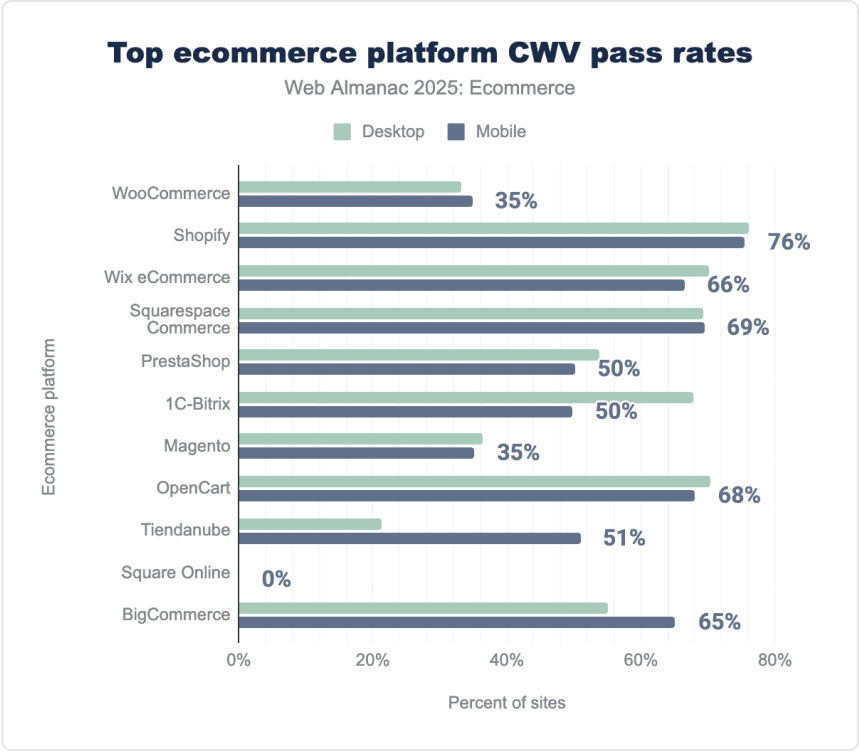


図13.9. 上位Eコマースプラットフォームの CWV 通過率。

プラットフォーム別CWV

プラットフォーム	オリジン数	良好なLCP	良好なINP	良好なCLS	良好なCWV
WooCommerce	394,462	45%	99%	68%	33%
Shopify	289,885	92%	99%	82%	76%
Squarespace Commerce	80,900	90%	100%	78%	69%
Wix eCommerce	55,706	77%	99%	91%	70%
PrestaShop	45,256	74%	99%	72%	54%
Magento	36,988	59%	99%	55%	36%
1C-Bitrix	31,150	86%	99%	80%	68%
OpenCart	14,452	87%	99%	80%	70%
Cafe24	13,661	98%	100%	46%	45%
BigCommerce	12,376	91%	99%	60%	55%

図13.10. 上位EコマースプラットフォームのCore Web Vitals良好率（デスクトップ）。

プラットフォーム	オリジン数	良好な LCP	良好な INP	良好な CLS	良好な CWV
WooCommerce	995,782	39%	88%	85%	35%
Shopify	567,932	86%	90%	92%	76%
Wix eCommerce	234,055	76%	85%	95%	66%
Squarespace Commerce	209,650	76%	96%	89%	69%
PrestaShop	87,486	65%	89%	81%	50%
1C-Bitrix	76,080	71%	71%	86%	50%
Magento	50,983	52%	87%	64%	35%
OpenCart	32,914	80%	93%	88%	68%
Tiendanube	21,836	60%	95%	84%	51%
Square Online	18,812	0%	39%	0%	0%

図13.11. 上位EコマースプラットフォームのCore Web Vitals良好率（モバイル）。

いくつかのパターンが繰り返し現れます：

- INPはほとんどの主要プラットフォームでデスクトップ全体で一般的に強く、現代のJSスタックとブラウザの改善が応答性を助けていることを示唆しています。
- LCPが最大の差別化要因です。高速なテーマと厳密に管理されたアプリエコシステムを持つプラットフォームはより良いスコアを獲得する傾向があります。
- WooCommerceは規模を持っていますが、自動的なスピードはありません：そのCWV通過率はSaaS重視のエコシステムよりも遅れており、これは無限のカスタマイズという性質と一致しています。

Lighthouse

Lighthouseは HTTP Archiveのラボベースの監査ツールです。Core Web Vitals（フィールドデータ）とは異なり、制御された環境（シミュレートされたデバイス、スロットルされたネットワーク/CPU）で実行され、パフォーマンス、アクセシビリティ、SEO、ベストプラクティスのスコアを生成します：

- **パフォーマンス**：Lighthouseパフォーマンスは制御されたテストプロファイル下での読み込みと応答性を要約するラボスコア（0〜100）です。プラットフォーム間の相対比較に最も役立ちます。
- **アクセシビリティ**：Lighthouseアクセシビリティは自動チェックに基づいています（すべてを捉えることはできません）が、ラベルの欠如、低コントラスト、不正確なセマンティクスなどの一般的な問題の有用なベースラインシグナルです。
- **SEO**：LighthouseのSEOスコアは技術的なSEOの基礎（例：タイトル/メタ、基本的なクロール可能性シグナル）を反映しています。これらのチェックは通過しやすいため、高い中央値が一般的です。
- **ベストプラクティス**：ベストプラクティスはセキュリティと信頼性のチェック（HTTPS、安全なJSパターン、最新のAPI）の集合体です。プラットフォームのデフォルトとテーマの品質を反映することが多いです。

Lighthouseはテストプロファイルを標準化するため、大規模なサイトセット間の比較に役立ちますが、フィールドデータはデバイス、ネットワーク、地域、ユーザー行動の実際の組み合わせを反映するため、実際のユーザーが体験するものと完全には一致しません。

プラットフォーム別Lighthouseスコアの中央値

プラットフォーム	全サイトの%	パフォーマンス（中央値）	アクセシビリティ（中央値）	SEO（中央値）	ベストプラクティス（中央値）
WooCommerce	7.10%	61	87	92	78
Shopify	4.32%	71	90	92	74
Squarespace Commerce	1.84%	65	94	92	96
Wix eCommerce	1.57%	88	96	100	78
PrestaShop	0.64%	59	78	92	81
1C-Bitrix	0.47%	58	75	92	59
Magento	0.41%	60	78	92	74
OpenCart	0.23%	63	79	91	78
Cafe24	0.20%	39	69	85	56
BigCommerce	0.15%	72	81	92	74

図13.12. 最も人気のあるEコマースプラットフォームのLighthouseスコア（デスクトップ）。

プラットフォーム	全サイトの%	パフォーマンス（中央値）	アクセシビリティ（中央値）	SEO（中央値）	ベストプラクティス（中央値）
WooCommerce	7.03%	38	87	92	79
Shopify	4.01%	55	91	92	75
Wix eCommerce	1.75%	64	95	100	79
Squarespace Commerce	1.56%	30	94	92	96
PrestaShop	0.59%	36	79	92	79
1C-Bitrix	0.53%	35	75	92	61
Magento	0.34%	34	79	92	75
OpenCart	0.23%	43	78	91	79
Tiendanube	0.15%	50	92	92	75
Square Online	0.14%	13	85	92	57

図13.13. 最も人気のあるEコマースプラットフォームのLighthouseスコア（モバイル）。

いくつかのハイレベルなパターン：

- SaaSストアフロントはパフォーマンスカテゴリーで高い方に集中する傾向があります（特にデスクトップで）。これはテーマとデフォルトのより厳密な管理と一致しています。
- アクセシビリティの中央値は上位プラットフォームで一般的に強いですが、中央値はロングテールの変動を隠す可能性があります。
- SEOとベストプラクティスのスコアはほとんどのプラットフォームで高い。チームが通常勝つか負けるかは、基本的な技術的SEOではなく、パフォーマンスと実装の規律です。

決済事業者

決済はEコマースが現実になる場所です。また、サードパーティのスクリプト、リダイレクト、不正防止ツール、コンプライアンスの制約という重大な依存関係の領域でもあります。

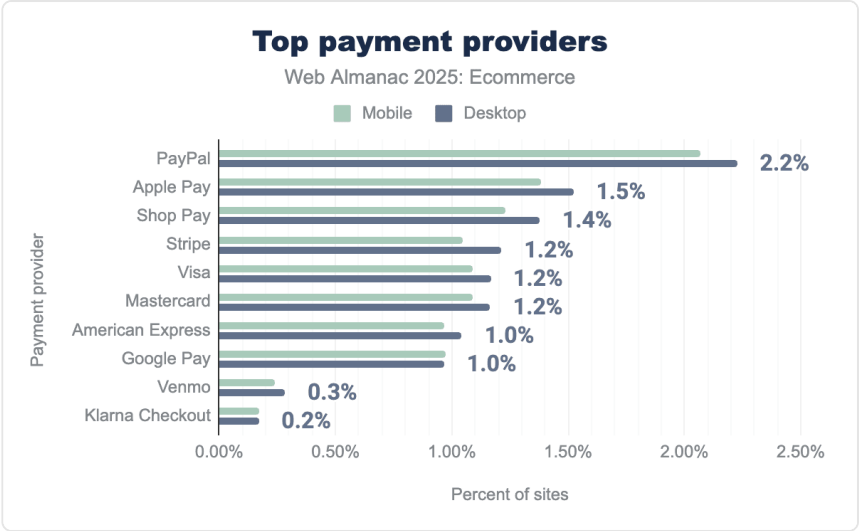


図13.14. 2025年のEコマースサイトにおける決済プロバイダーの分布。

上位10の決済技術が両デバイスでの決済検出の大部分を占めており、決済が素早く統合されるという改めての証しです。ただし、単一または2つのプロバイダーが全体的に支配的として際立っているわけではないことは注目に値します。

2022年以降の変化は？

最も注目されるトレンドは、PayPalの決済検出のシェアが低下し、ウォレットとプロセッサーファーストのエコシステムが成長していることです：

- PayPalは2022年の決済検出の約39%から2025年の約30%に低下しています（デスクトップとモバイルの両方でこのパターンが見られます）。
- StripeとGoogle Payは同期間にシェアを獲得しています。
- Apple PayとShop Payは高いレベルで比較的安定しています。

これはPayPalが消滅しているという意味ではありません。ネイティブウォレット、リンクベースのチェックアウト、プラットフォームネイティブのアクセラレーターが標準になるにつれ、特に決済レイヤーがより多様化しているという意味です。

地域別決済事業者

83か国中70か国

図13.15. PayPalが上位の決済プロバイダーである国の数。

- モバイルでは、PayPalは83の地域のうち70の地域でトップの決済プロバイダーです。
- デスクトップでは、リーダーシップはより分散しています：Stripeが31の地域でリードし、PayPalは22でリードしています。

注目すべき例：

- Apple Payはニュージーランドでリードしています（デスクトップとモバイルの両方）。
- Braintreeは台湾で際立っています。
- いくつかのアフリカと中東の市場では、このデータセットでStripeが最も一般的なトッププロバイダーとして示されています（例：ナイジェリア、ケニア、UAE）。

結論

2025年のEコマースは集中していると同時に多様でもあります。少数のプラットフォームが検出されたEコマースサイトの大部分を占めており、WooCommerceとShopifyが先頭に立っている一方、地域とニッチなシステムのロングテールは特定の市場で引き続き重要です。ウェブサイトのランクは別のレイヤーを追加します：エンタープライズ向けのプラットフォームはより高いトラフィックティアで不釣り合いに登場し、ロングテールのサイトは採用しやすくコストの低いソリューションに傾く傾向があります。

パフォーマンスは注釈ではなく、差別化要因であり続けます。フィールド指標（Core Web Vitals）とラボ監査（Lighthouse）の両方が、より厳密なプラットフォームコントロールがより良い中央値の結果と相関する可能性があることを示していますが、そのギャップは運命ではありません。エンジニアリングの規律が強い場合、セルフホスト型と大幅にカスタマイズされたスタックも良いパフォーマンスを発揮できます。決済技術も素早く統合されます：少数のプロバイダーが検出を支配し、ウォレットとプロセッサーファーストのエコシステムはシェアを獲得し続けています。

Eコマースの次の章は、どのプラットフォームかだけでなく、どのチャネルかです：音声、ライブコマース、エージェンティックコマースがストアをより速く、よりアクセシブルで、より機械的に消費可能にすることを促しています。勝者は、カタログの品質、パフォーマンス、信頼を製品機能として扱う人たちです。なぜならますます、買い物客はページをクリックして回る人間ではないかもしれないからです。

著者

Amandeep Singh

@amanvirdi amandeepsinghvirdi amandeepsinghvirdi https://byaman.com/

Amandeep Singhは2009年からウェブ開発に携わり、フロントエンド開発、UI/UX、Shopify、BigCommerce、WordPress、プログラミングについてbyaman.com[455]でブログを書いています。ライター、メンター、スピーカーとして活動しています。

455. https://byaman.com/

部 IV 章 14
ページの重さ

Richard Barret と Jamie Indigo によって書かれた。
Dave Smart によってレビュ と による分析。
Montserrat Cano 編集。
Sakae Kotaro によって翻訳された。

はじめに

ウェブの黎明期、すべてのバイトは贅沢品でした。開発者はGIFのディザリングマッピングやスクリプトの手動最適化に何時間も費やし、56kモデムでページが閲覧できるようにしていました。今日、ギガビット光ファイバーと5G全盛の時代に、その希少性の考え方はほとんど消え去りました。しかし「帯域幅のパイプ」が広がるにつれ、私たちがそこに送り込むコンテンツもそのスペースを埋めるために膨張しています。

ページの重さ（ユーザーのデバイスに転送されるバイトの総量）は、ウェブの健全性を理解するための最も重要な指標の一つであり続けています。数メガバイトの追加をモダンなリッチエクスペリエンスの些細なコストと見なしたくなりますが、実際はずっと複雑です。

ページの重さとは？

ページの重さ（ページサイズとも呼ばれる）とは、ユーザーが特定のウェブページを表示す

るためにダウンロードしなければならないデータの総量で、キロバイト（KB）またはメガバイト（MB）で測定されます。

URLにアクセスすると、ブラウザは1つのファイルをダウンロードするだけではありません。ページを正しく表示・機能させるために必要なさまざまなアセットに対して、数十または数百のリクエストを送信します。これらすべての「送信された」バイトの合計がページの重さを構成します。

現代のウェブページは複数の異なるタイプのリソースの組み合わせです。それぞれがサイトの合計「重量」に貢献します：

- **画像・動画：** 高解像度の写真、バックグラウンド動画、GIFはすぐにページサイズを膨らませます。
- **JavaScript：** インタラクティビティを提供するファイル（メニュー、トラッキングピクセル、アニメーションなど）。KBではしばしば画像より小さいですが、JavaScriptはブラウザが解析・実行のために多大なCPUパワーを費やす必要があるため「重い」です。
- **CSS：** ページのレイアウト、色、フォントを決定するスタイルシート。
- **フォント：** 複数のウェイト（太字、斜体、細字）が使用される場合、カスタムウェブフォントは数百KBを追加する可能性があります。
- **HTML：** ページの構造的なコードで、通常は合計重量の最も小さい部分です。
- **サードパーティスクリプト：** 他のサーバーから取得される広告、アナリティクス、ソーシャルメディアウィジェットが含まれます。

すべてのバイトが平等に作られているわけではありません。本章では、ファイルタイプのバイト数とリクエスト量によってページの重さを探ります。

なぜ重さが重要か

ページの重さはパフォーマンスとアクセシビリティの直接的な指標です。「重い」ページはいくつかの負の連鎖反応を生み出します：

1. **パフォーマンスギャップ：** より大きなペイロードは解析とレンダリングのためにより多くのCPUサイクルを必要とし、より多くのデバイスメモリを使用します。これは接続速度に関わらず、ローエンドデバイスでの遅い体験につながることが多いです。
2. **経済的な負担：** 世界の多くの地域でデータは従量制の商品です。5 MBのページは単に遅い体験ではなく、排他的なものです。ユーザーがページを読み込むため

のデータを負担できない場合、サイトは定義上そのユーザーにとってアクセスできないものになります。

3. **アクセシビリティの障壁：** ページが「重い」と、ただ読み込みが遅くなるだけでなく、物理的・認知的に使いにくくなります。過度なページ重量は重大な不平等を生み出し、非力なデバイスや高価で低速な接続でデータキャップが限られているユーザーを著しく不利にします。ページの重さが障害を持つ何百万人ものユーザーへの静かだが重大な参入障壁となる方法についての詳細は、アクセシビリティチャプターを参照してください。
4. **環境への影響：** 転送されるすべてのメガバイトはエネルギーを必要とします。データセンターから冷却システム、ユーザーの手元のデバイスまで。ウェブが成長するにつれ、そのカーボンフットプリントも増大します。
5. **速度とSEO：** 重いページは特に遅い接続では読み込みに時間がかかります。GoogleはコアアルゴリズムにCore Web Vitalsを通じたページ速度を使用しており、膨らんだページは検索結果で低くランク付けされる可能性があります。SEOチャプターも参照してください。

重量効果は3つの主要カテゴリーに分けることができます：ストレージ、送信、レンダリング。

ストレージ

ストレージとは、アセット（画像、スクリプト、HTML）がウェブサーバーまたはCDN上にどのように存在するかを指します。この段階では、ページの重さはディスク上のファイルサイズに関するものです。

- **静的圧縮：** 開発者はしばしば高度に圧縮された形式でファイルを保存します（画像のWebPやBrotli圧縮テキストなど）。1 MBのファイルは300 KBとして保存できます。
- **データベースのボトルネック：** 動的サイトの場合、「重さ」はデータベースクエリから始まります。サーバーが1ページを生成するためにデータベースから2 MBの生データを取得しなければならない場合、単一のバイトも送信される前に初期レスポンスタイム（Time to First Byte (TTFB)[456]）が増加します。
- **コスト：** 非効率なストレージは速度だけでなく、ホスティングコストとデータセンターのカーボンフットプリントを増加させます。

456. *https://web.dev/articles/ttfb*

送信

送信とは、保存されたファイルをインターネット上で移動するプロセスです。ここでネットワークの制約がページの重さをパフォーマンスの障壁に変えます。

- **転送サイズと実際のサイズ：** 「オンザフライ」圧縮（Gzipなど）のおかげで、ネットワーク上で送信されるバイト数は元のファイルサイズよりもはるかに小さいことが多いです。
- **レイテンシとラウンドトリップ：** 問題はどれだけのデータが送信されるかだけでなく、何枚のファイルが送信されるかです。それぞれの独立したファイルはサーバーへの「ラウンドトリップ」を必要とします。50枚の小さな画像（合計1 MB）のページは、50件の独立したリクエストの転送オーバーヘッドにより、実際には1枚の大きな2 MBの画像を持つページよりも遅く感じることがあります。
- **ボトルネック：** モバイルの4G/5Gでは、信号干渉が「パケットロス」を引き起こす可能性があります。ページが重いほど、パケットがドロップしてブラウザが再度要求するよう強制し、目に見えるハングを引き起こす可能性が高くなります。

レンダリング

レンダリングとは、データが到着した後に起こることです。これはページの重さの最も誤解されている部分です。ファイルがダウンロードされると、デバイスによって「解凍」されて処理される必要があります。

- **メモリ膨張：** 画像は転送中に200 KBしか占有しないかもしれませんが、ブラウザが「レンダリング」すると、デバイスのRAMでの生のピクセルにデコードされなければなりません。その200 KBのファイルは5 MBのメモリを簡単に占有する可能性があります。
- **JavaScriptの重税：** これはレンダリングの最も「重い」部分です。100 KBの画像は単なるピクセルですが、100 KBのJavaScriptは作業です。CPUはそのコードを解析、コンパイル、実行しなければなりません。ローエンドのスマートフォンでは、この「重さ」は画面を数秒間フリーズさせる可能性があります。
- **DOMの複雑さ：** すべてのHTMLタグはブラウザのメモリに「ノード」を追加します。5,000ノードを持つページ（「重い」DOM）は、インターネット接続の速さに関わらず、スクロールがもたつく感じにさせます。

ウェブサイトはレンダリング戦略を変更しつつあります。これはAIチャットボットや他の大規模言語モデルツールの増大によってウェブサイトがどのようにアクセスされ消費されるかを再考することで促されることもあります。これらの変更のすべてがAIクローラーのアクセ

シビリティを目的としているわけではありません。AIクローラーのアクセシビリティの技術要件を特定するためのテストが行われており、これらはクローラー間で異なる場合があります。AIクローラーはJavaScriptをレンダリングしない[457]ため、すべての重要な情報は最初の生のHTMLに存在しなければならないことが分かっています。

これらの要因への認識が高まることで、JavaScriptファイルの使用削減を含む、このような戦略のより広い採用が将来のエディションで期待できます。

私たちは何を送っているのか？

本章では、数百万のウェブサイト全体でトリリオンバイトを分析して、根本的な質問に答えます：私たちはペイロードからより多くの価値を得ているのか、それとも単に「重量クリープ」に屈しているのか？メディア（画像と動画）の優勢を探ります。これらは転送バジェットのライオンシェアを主張し続けており、JavaScriptの着実な台頭はファイルサイズが示唆するよりもはるかに重いパフォーマンスの税金を運んでいます。

リクエストバイト

時系列でのページの中央値の重さは、ページサイズの成長が加速していることを示しています。2024年10月以降、特にモバイルデバイスで顕著な上昇トレンドがあります。これは、平均して内部ページよりも約45.8%大きいホームページでより大きなスケールで示されています。

ホームページのページ重量も同様に加速していますが、それより早く2023年初め/2022年後半頃に起きています。

これにはより広い影響があります：

AIへのページ重量の影響

エネルギー消費と計算コストが増加しており、重いページは訪問ごとにより多くの帯域幅、レンダリング、CPU負荷を意味し、これらすべてがウェブクロールとインデックスのエネルギーフットプリントを増加させます。

これは、エンティティ抽出やコンテンツ要約などのアクションを実行してウェブを「理解」しようとするAI駆動のクローラーが計算予算を管理するためにクロールの深さや頻度を制限する可能性があるため、より遅いまたは浅いクロールをもたらします。

これは、解析してモデル化しやすいため、読み込みが速いまたはより意味的に明確なサイト

457. *https://vercel.com/blog/the-rise-of-the-ai-crawler*

など、より軽いフットプリントのウェブコンテンツへのバイアスにつながる可能性があります。JavaScriptとクライアントサイドレンダリングで過負荷になったサイトは、AIモデルが採用するクローラーはレンダリングしないため、大規模なトレーニングセットで過小代表になる可能性があります。 最終的に、パフォーマンスがより重要になるにつれ、AIは遅いまたは非常に重いページで見るものをスキップまたは切り詰める可能性があります。

ユーザーへのページ重量の影響

ページが重くなるにつれ、より多くのサイトがLCPとINPで基準を下回る可能性があり、これは検索結果での可視性を直接低下させる可能性があります。さらに重要なのは、利便性がますます重要な要素となっているため、これはフラストレーティングなユーザーエクスペリエンスを生み出し、コンバージョン率の低下につながる可能性があります。

重いページは、これらの増加によって不釣り合いに影響を受ける低速接続またはローエンドデバイスのユーザーとの間のデジタルデバイドを広げます。これは、検索マーケターにとってこれらのユーザーを念頭に置いてウェブサイトを最適化することの重要性を浮き彫りにしています。

パブリッシャーへのページ重量の影響

パフォーマンスの低下はオンライントラフィック、コンバージョン率、広告ビューの低下と相関しています。広告テックとアナリティクスのスクリプトは追加の重量を生み出し、ユーザーエクスペリエンスと収益化の間に摩擦を生み出します。

まとめると：AIシステムは、スクレイピングを減らして要約を増やし、ギャップを「幻覚」で埋めることで適応するかもしれません。これはブランドへのユーザーの信頼を低下させ、オンライントラフィックとコンバージョンの低下につながる可能性があります。ユーザーの行動は、利便性が人々のオンライン情報へのアクセス方法において重要な役割を果たし続ける中で、従来のページ訪問からAI介在のブラウジング体験へとシフトする可能性があります。

経年ページ重量

Web Almanacの設立以来、私たちは一貫して2つのトレンドを観察してきました：

1. リクエストの総量が増加する。
2. デスクトップのページ読み込みはモバイルよりも多くのリクエストをもたらす。

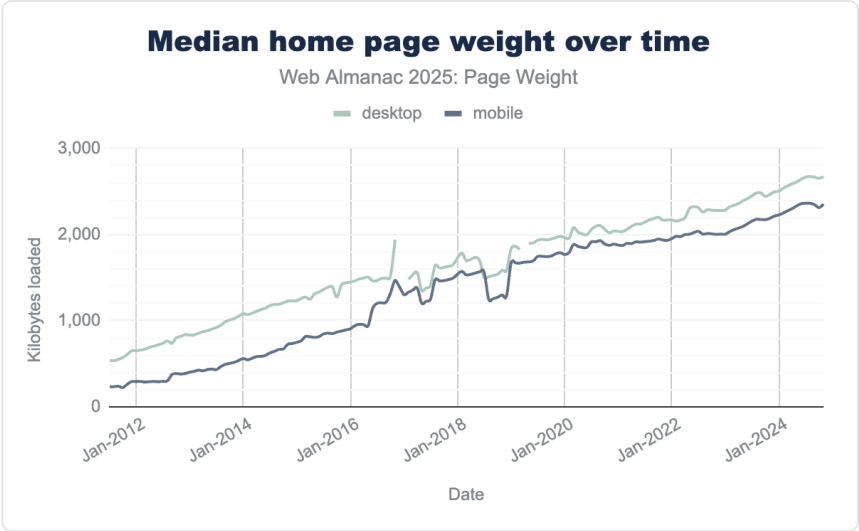


図14.1. 時系列で見た中央値ホームページの重量。

2015年7月、モバイルホームページの中央値はわずか845 KBでした。2025年7月時点で、同じ中央値ページは2,362 KBになっています。過去10年間で202.8%の増加をもたらしました。デスクトップホームページの中央値は同期間に110.2%増加しました。

前年比で、ホームページサイズの中央値は7.8%増加して2.7 MBになりました。モバイルホームページの中央値は2.6 MBで、2024年の2.4 MBから8.4%増加しています。2025年、デスクトップページサイズの中央値は2.9 MBに達し、2024年の中央値2.7 MBから7.3%増加しました。

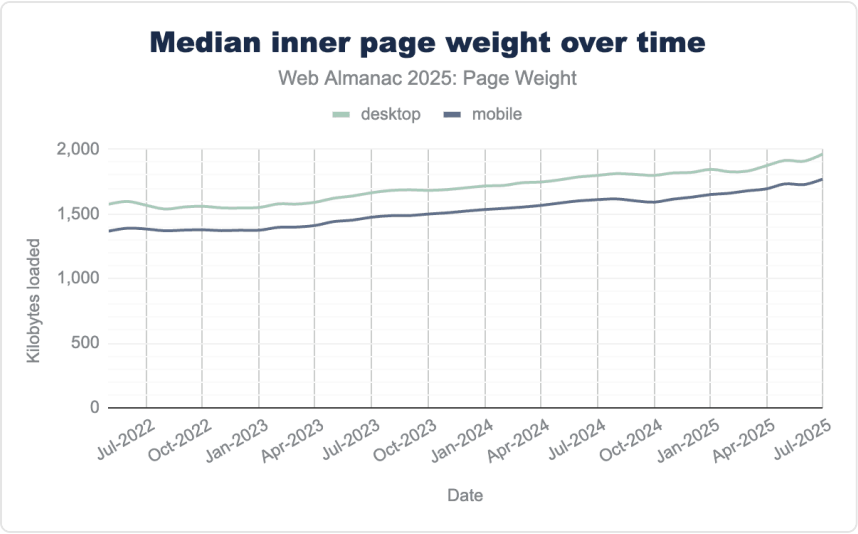


図14.2. 時系列で見た中央値内部ページの重量。

内部ページの重量は前年比9.5%増加し続けました。2022年から内部ページの重量追跡データが開始されて以来、モバイル内部ページの中央値の重量は27.8%増加して1.8 MBになり、デスクトップは同期間に25.2%増加して2 MBに達しました。

バイト数で見るページ重量

ページ機能のすべての更新は、コア機能を維持しながらパフォーマンスを向上させるために設計された追加の重量とファイルタイプをもたらします。一般的なファイルタイプ、その頻度、およびレスポンスサイズを調査し、異なるデバイスとページタイプ間の比較を含む実装のより明確な理解を得ました。

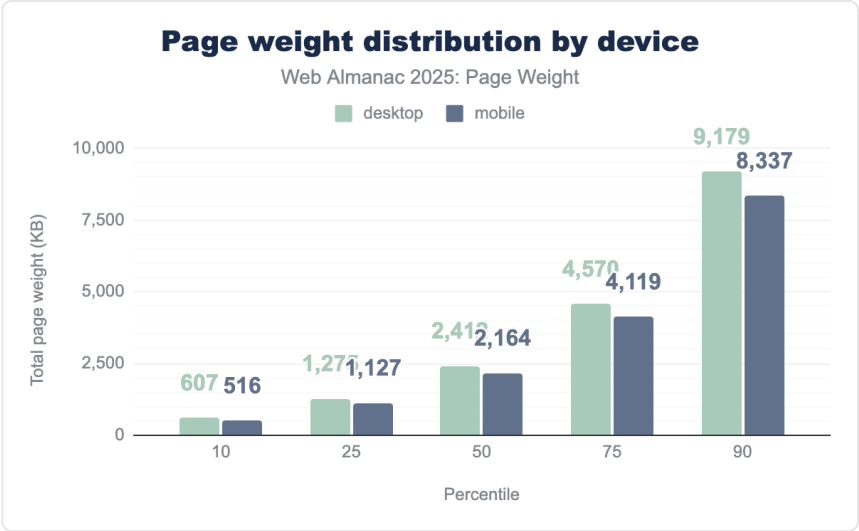


図14.3. デバイス別ページ重量分布。

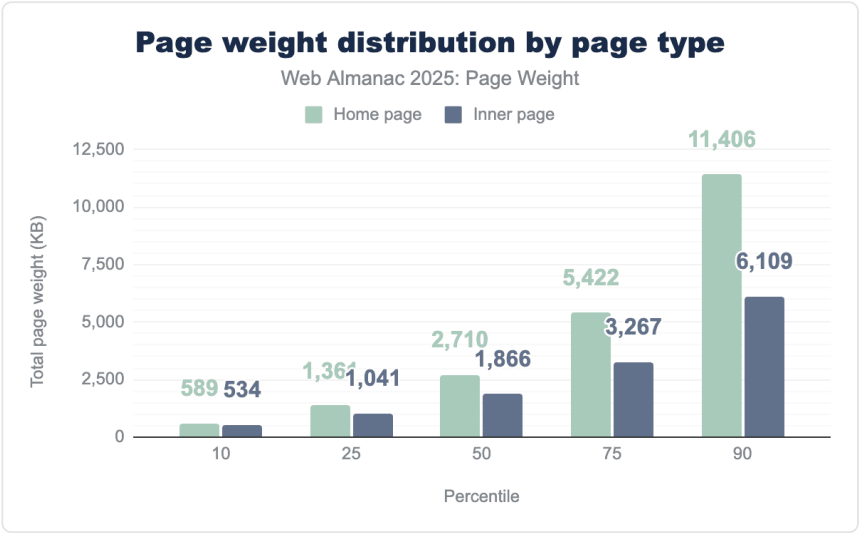


図14.4. ページタイプ別ページ重量分布。

バイト数で見た中央値ページ

ウェブをよりよく理解するために、第50パーセンタイルを調べることができます。中央値は典型的な値を表し、相対的なページ重量を研究するためのコンテキストを提供します。

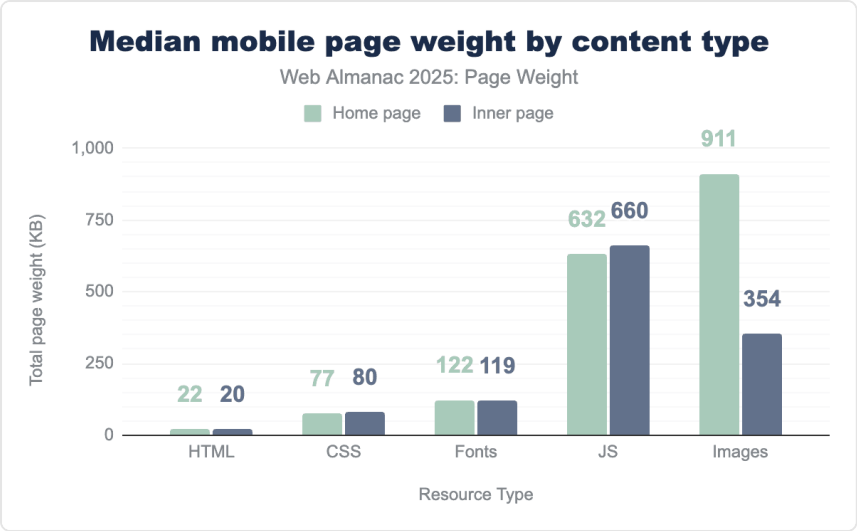


図14.5. コンテンツタイプ別モバイルページの中央値の重量。

2025年、モバイルホームページの中央値は以下を使用しました：

- 22 KB のHTMLリソース
- 77 KB のCSSリソース
- 122 KB のフォント
- 632 KB のJavaScript
- 911 KB の画像

内部ページは同様でしたが、画像については911 KBから354 KBに減少しました。

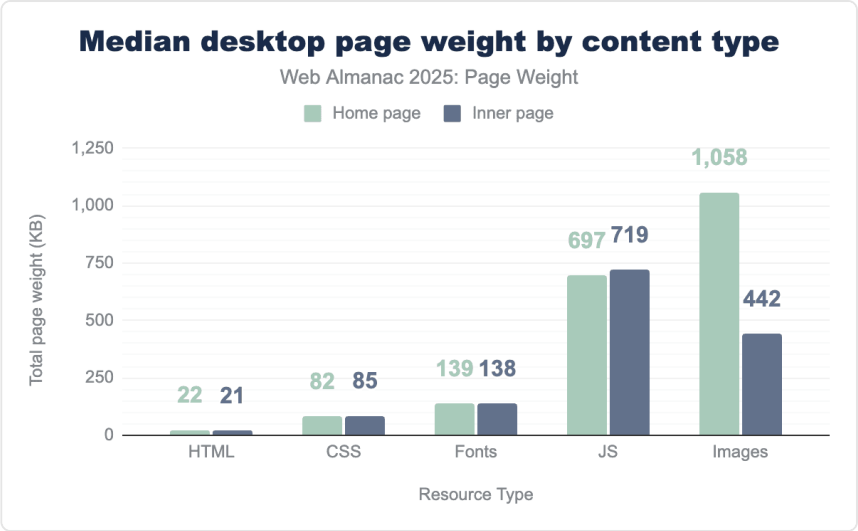


図14.6. コンテンツタイプ別デスクトップページの中央値の重量。

デスクトップページの中央値のリソースタイプ別重量はモバイルと一致しましたが、リソースはKBでわずかに大きかったです。2025年、デスクトップホームページの中央値は以下を使用しました：

- 22 KB のHTMLリソース
- 82 KB のCSSリソース
- 139 KB のフォント
- 697 KB のJavaScript
- 1,058 KB の画像

両デバイスタイプで、画像はキロバイト数で最大の差を示し、ホームページで1.06 MB、内部ページで442 KBが使用されました。

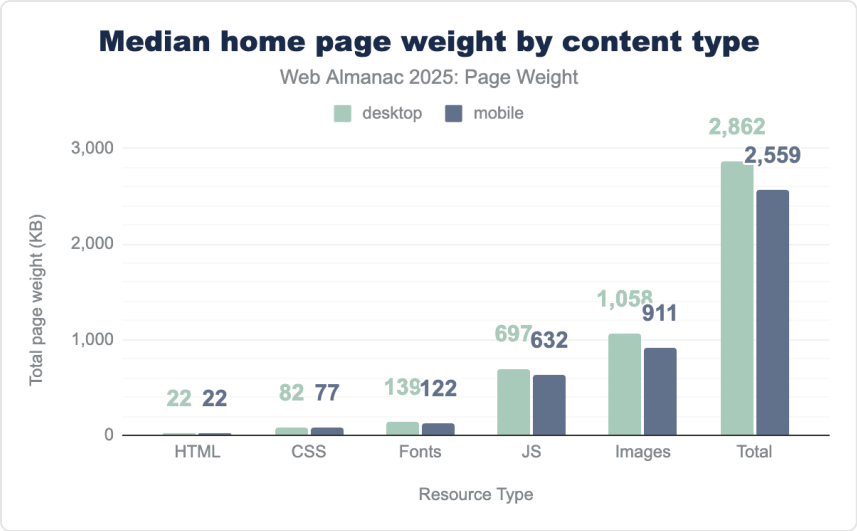


図14.7. コンテンツタイプ別ホームページの中央値の重量

2025年のホームページの中央値はデスクトップで2.86 MB、モバイルで2.56 MBでした。画像はモバイルとデスクトップの両方で最も多くのバイトを占め、次いでJavaScriptとフォントが続きました。

コンテンツタイプ別レスポンスサイズ

すべてのファイルタイプが同等に作られているわけではありません。また、同じサイズでもありません。ファイルタイプはデータがどのように保存・エンコードされるかを定義し、プログラムにファイルの内容を開いて表示する方法を伝えます。

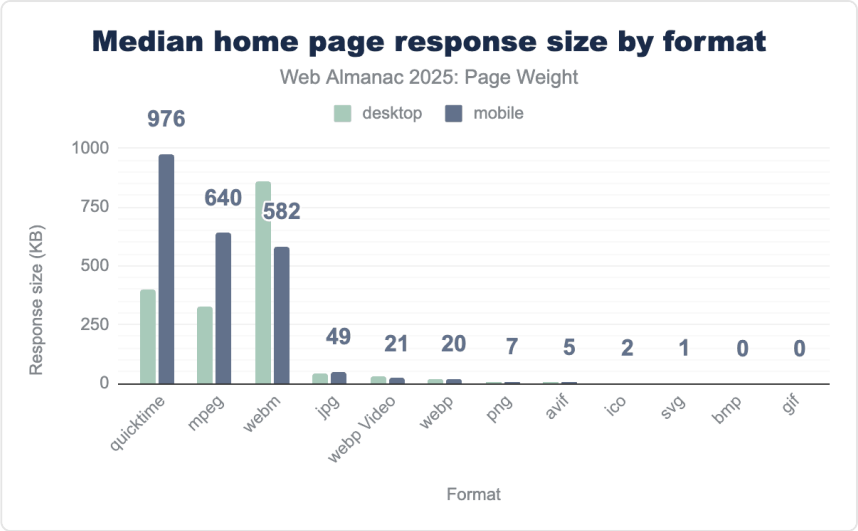


図14.8. フォーマット別ホームページの中央値のレスポンスサイズ。

モバイルデバイスは一般的に、パフォーマンスの高い体験を提供することがより難しいです。より小さく多様な画面サイズと弱い接続のため、コンテンツの読み込みに時間がかかる可能性があります。これが、モバイルホームページが最も重いファイルの一部を持っていたことを奇妙にさせています。

最大のファイルタイプ、QuickTimeとMPEGは動画に使用されています。モバイルホームページの中央値は976 KBのQuickTimeを使用しており、これはデスクトップの400 KBより244%多いです。同様に、モバイルホームページの中央値はデスクトップよりも196%多くのMPEGバイトを使用しました。3番目に大きいファイルタイプはWebMで、より速い読み込みと低帯域幅使用のために最適化された動画フォーマットです。モバイル接続に理想的であるにもかかわらず、デスクトップでより多くのWebMバイトが送信されました。

さらに調査したところ、動画ファイルがブラウザが異なるRangeヘッダーで複数のリクエストを行うような重複ダウンロードに特に影響を受けやすいことに驚きました。これにより動画のバイト数が膨らみ、モバイルでより悪化しているようでした（おそらく遅いネットワークダウンロードによる）。この動作についてのドキュメントは見つかりませんでしたが、読者の方が説明できる場合はぜひ聞いてみたいと思います。

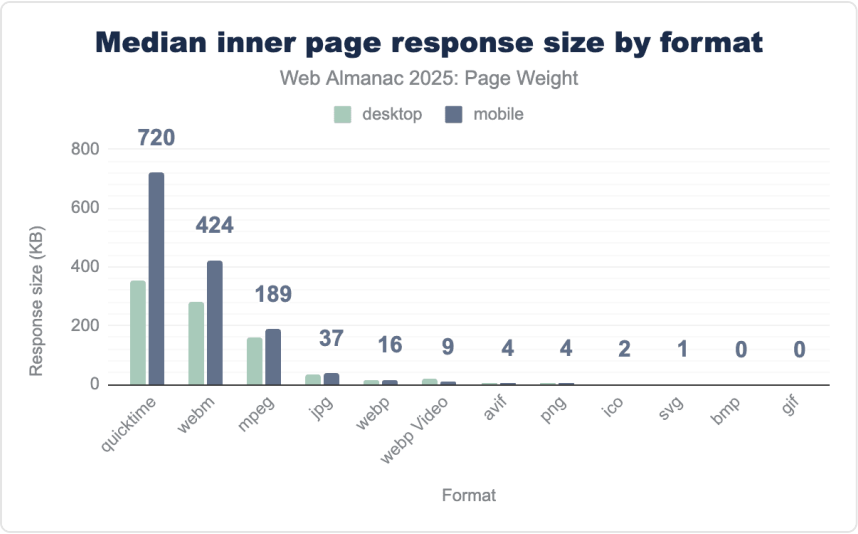


図14.9. フォーマット別内部ページの中央値のレスポンスサイズ

内部ページの中央値では、QuickTimeファイルがバイト数で最大のファイルタイプであり続けました。モバイルは720 KBで、デスクトップの352.6 KBの202%でした。WebMは内部ページで2番目に大きいファイルタイプとしてMPEGを上回り、モバイルで423.6 KB、デスクトップで280.8 KBでした。次いでMPEGが続き、ホームページよりもデバイスタイプ間でより均等な分布（デスクトップ161.6 KB；モバイル188.8 KB）を示しました。

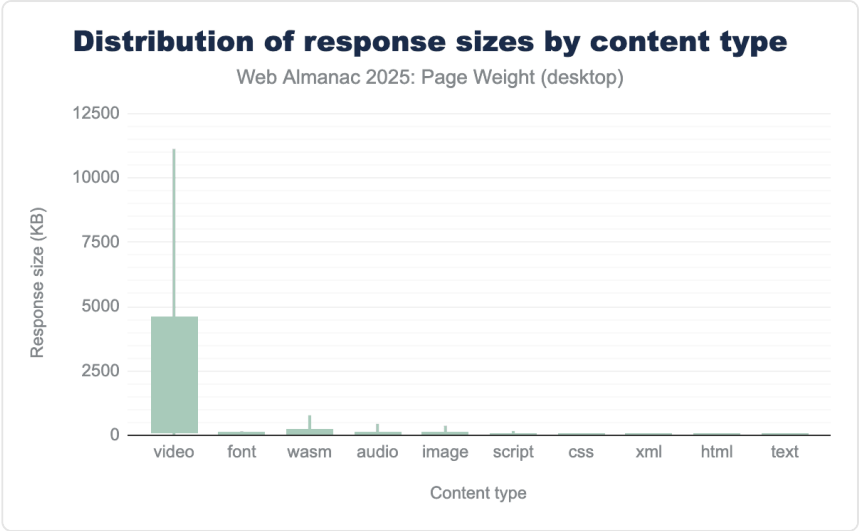


図14.10. コンテンツタイプ別レスポンスサイズの分布。

動画は本来、最も大きなファイルになりがちです。また、サイズの範囲も最大で、第10パーセンタイルの7 KBから第100パーセンタイルの1GBまであります。その範囲は、第10パーセンタイルで0 KB、第100パーセンタイルで1.2GBを示した画像ファイルタイプのみによって上回られます。XMLファイルは最も小さい範囲を示し、第10から第50パーセンタイルまでが0 KBで、最も極端なデスクトップページは第100パーセンタイルで52.1 MBでした。

画像バイト

画像バイトとは、写真、アイコン、イラストなどのビジュアル要素をレンダリングするために必要なバイナリデータのことです。HTMLやCSS、JavaScriptのようなテキストベースのファイルとは異なり、非常に効率的で容易に圧縮できますが、画像データは本来密度が高いです。これらのバイトの「重さ」は3つの主要な要因によって決まります：

1. **解像度：** アセットの総ピクセル数。
2. **エンコーディング：** データを保存するために使用される数学的方法（例えば、JPEG、PNG、またはWebPやAVIFのような現代的な形式）。
3. **圧縮：** ファイルサイズを縮小するために冗長なデータを除去する程度で、しばしば視覚的な忠実度のコストを伴います。

画像バイトの蓄積は「重い」ページを生み出し、これはレイテンシの増加とLargest Contentful Paint（LCP）スコアの低下と直接相関しています。モバイルデバイスや帯域幅が制限された接続のユーザーにとって、高い画像バイト数は法外な読み込み時間とデータコストの増加につながる可能性があります。

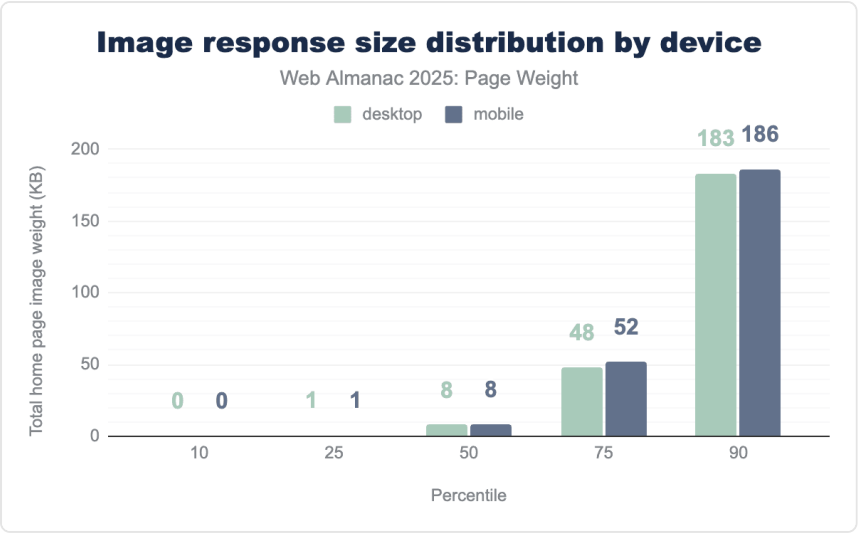


図14.11. デバイス別画像レスポンスサイズの分布

第50パーセンタイルでは、個々の画像ファイルはモバイルとデスクトップの両方で8 KBでした。多くのサイトがアナリティクスの一部としてトラッキングピクセルとして知られる小さな画像を使用していることは注目に値します。これらの小さな、しばしば見えない1x1ピクセルの画像はウェブページに埋め込まれ、読み込まれるとサーバーリクエストをトリガーします。これらのファイルは標準的なユーザー向け画像アセットと区別されないため、中央値の画像サイズが予想より小さく見える原因となる可能性があります。

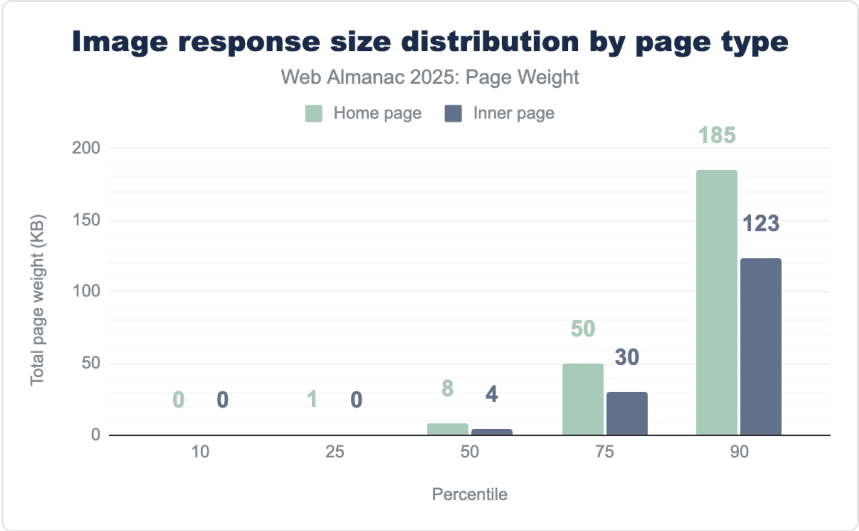


図14.12. ページタイプ別画像レスポンスサイズの分布。

内部ページはホームページよりも個々の画像ファイルサイズが小さく、中央値は4 KBでした。第90パーセンタイルでは、ホームページ画像は185 KB、内部ページ画像は123 KBでした。

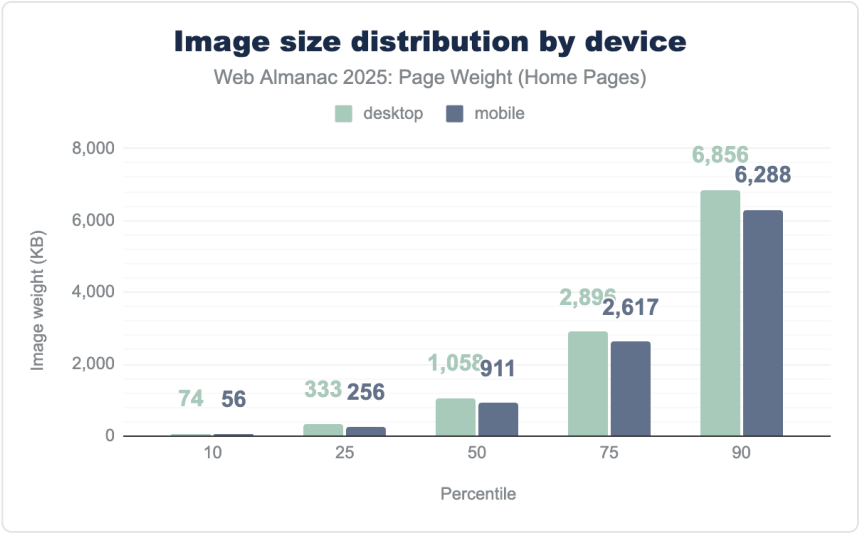


図14.13. デバイス別画像サイズの分布

ウェブページが単一の画像しか使用しないことはほとんどありません。まとめると、デスクトップページの中央値は1,059 KBの画像を使用しました。モバイルページは911 KBを使用しました。デスクトップページは一貫してより多くの合計画像バイトを読み込みました。

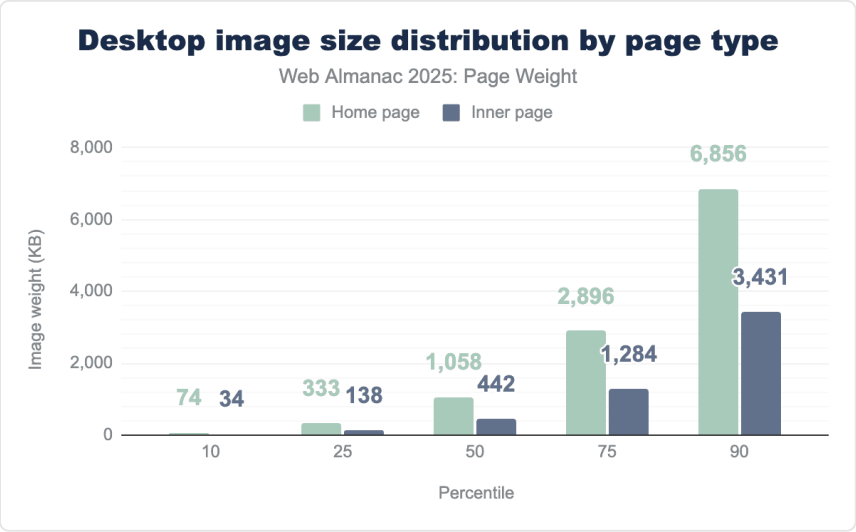


図14.14. ページタイプ別デスクトップ画像サイズの分布。

デスクトップホームページは一貫して内部ページよりも多くの画像を使用しました。ホームページの中央値は類似した内部ページの239%の画像バイトを使用しました。第90パーセンタイルでは、ホームページは内部ページの3,431 KBと比較して6,856 KBとほぼ2倍の画像バイトを使用しました。

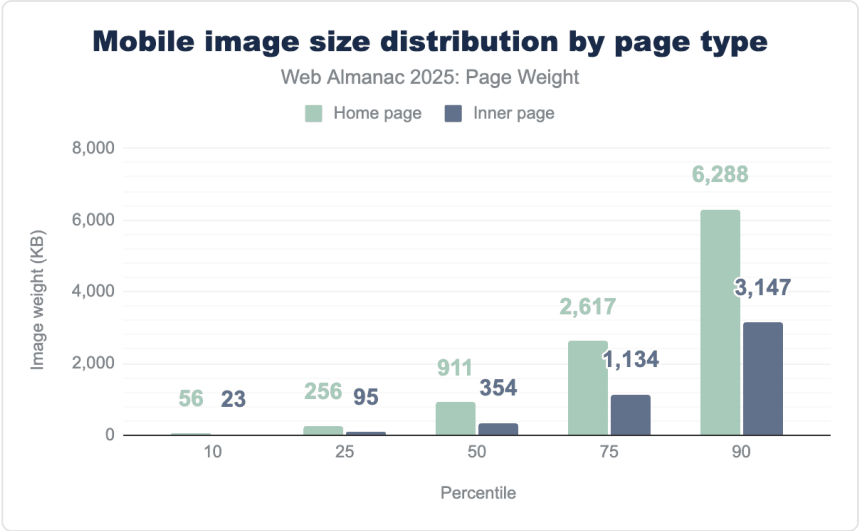


図14.15. デバイス別モバイル画像サイズの分布

内部ページとホームページの画像使用の範囲をモバイルデバイスのみに絞っても、トレンドは続きます。モバイルデバイスはデスクトップと比較して接続速度が遅く、計算能力が限られていることが多いです。しかしこれはモバイルホームページが画像バイトでそのトピックを表現することを妨げないようです。モバイルホームページの中央値は911 KBの画像を使用し、内部ページは354 KBを使用しました。第90パーセンタイルでは、モバイルホームページは6,288 KBの画像を使用し、デスクトップの6,856 KBにほぼ匹敵しました。

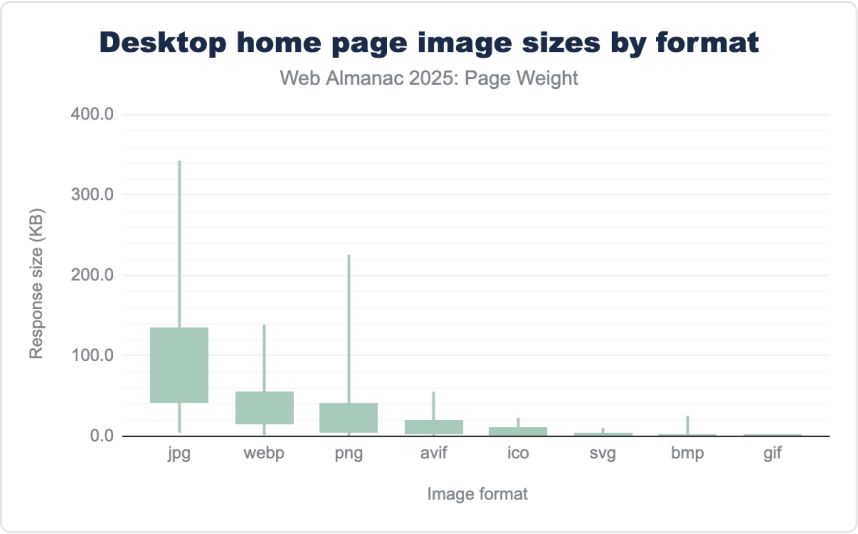


図14.16. フォーマット別デスクトップホームページ画像サイズの分布。

ファイルタイプが画像サイズに与える影響は、デジタル情報がどのように整理されて破棄されるかにかかっています。すべての画像は生のピクセルのグリッドとして始まりますが、ファイルフォーマットがそれらのピクセルを保存する方法によって、ファイルが数キロバイトか数メガバイトかが決まります。

- **JPG** ファイルは非可逆型のファイルタイプです。大幅なサイズ削減を実現するために「不要な」データを永久に削除します。どれだけ削除されるかはファイル作成者の裁量によります。これがJPGファイルサイズが最大の範囲にまたがる理由で、第10パーセンタイルの3 KBから第90パーセンタイルの278 KBまでの範囲があります。

- **WebP** は画像の汎用ファイルタイプと考えられています。非可逆と可逆の両方として利用可能で、これらのファイルタイプは第10パーセンタイルの1 KBから第90パーセンタイルの97 KBまでの範囲にまたがります。

- **AVIF** は広く採用されている画像ファイルフォーマットの中で最も新しく、最先端の圧縮効率のために2019年に導入されました。このファイルフォーマットは高品質なウェブ画像を生成しながら、非可逆と可逆の両方の圧縮に使用できます。AVIFは第10パーセンタイルの1 KBから第90パーセンタイルの37 KBまでの

範囲にあります。

- **PNG** は大きくても完璧な画像ファイルを作成するために使用され、ロゴやアイコンに理想的なフォーマットです。PNGは第10パーセンタイルの0 KBから第90パーセンタイルの278 KBまでの範囲にあります。

JavaScriptバイト

JavaScriptの重さの増大は、ウェブサイトがインタラクティブ機能を提供するためにJavaScriptに頼り続ける中でのもう一つの重大なトレードオフです。これらのスクリプトはユーザーエンゲージメントを向上させますが、ペイロードサイズの継続的な増加はサイトパフォーマンスへの課題となっています。

このデータセットは圧縮されたJavaScriptファイルのバイトを表していることに注意することが重要です。圧縮の効率によって、これは大きな違いになることがあります。転送量が大きいほど（圧縮を仮定して）、非圧縮と圧縮の差は指数関数的になる可能性があります。さらに、HTMLページにインライン化されたJavaScriptはこのデータには含まれていません。

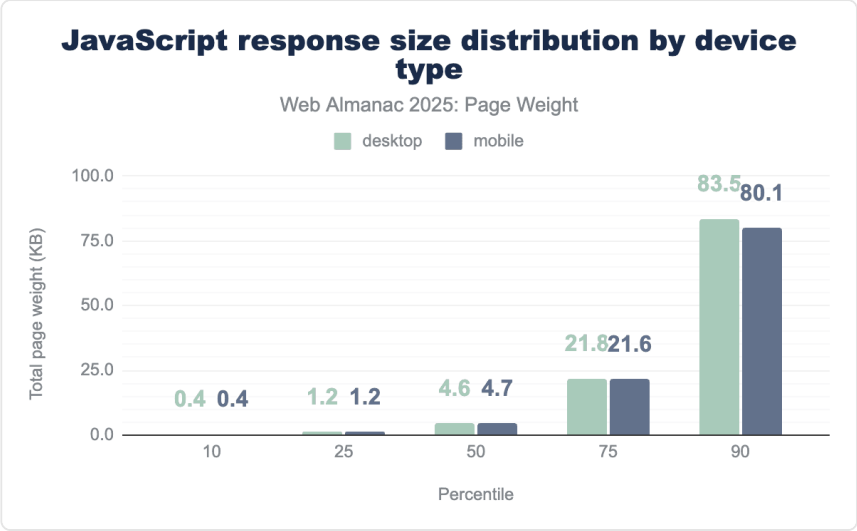


図14.17. デバイス別JavaScriptファイルサイズの分布。

どんな雨粒も洪水を引き起こしたとは思わないように、JavaScriptファイルの個々の中央値はデスクトップで4.6 KB、モバイルで4.7 KBと非常に小さいです。小さなJavaScriptファイルは、ユーザーインタラクションへのレスポンスを遅らせることなく小さなタスクを完了で

きるため理想的です。第10パーセンタイルでは、最も軽いJavaScriptファイルは両デバイスタイプで0.4 KBでした。第90パーセンタイルでは、デスクトップのJavaScriptファイルは83.5 KB、モバイルは80.1 KBでした。

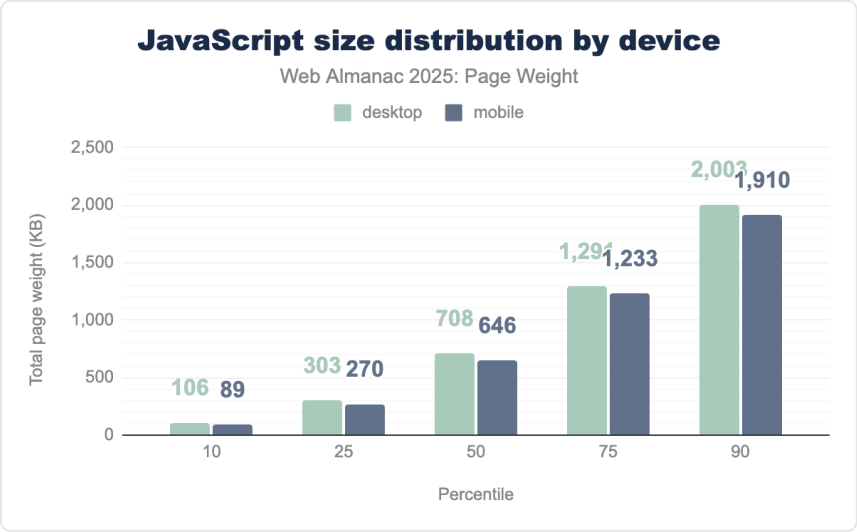


図14.18. デバイス別JavaScriptサイズの分布。

比較的小さな個々のJavaScriptファイルをまとめると、全体的なページでの使用状況をより良く把握できます。JavaScriptの使用はモバイルとデスクトップ間で比較的一貫していました。第10パーセンタイルでは、デスクトップは106 KB、モバイルは89 KBを使用しました。デスクトップページの中央値は708 KBのJavaScriptを使用しています。これはモバイルの646 KBよりわずかに高いです。これらの数字は中央値を超えて急速に膨らみます。第90パーセンタイルでは、デスクトップは2 MBをわずかに超え、モバイルは1.9 MBを使用しました。

第100パーセンタイルのデスクトップページは189.9 MBのJavaScriptを使用し、モバイルは181.1 MBを使用しました。比較のために、標準解像度の30分のテレビ番組をダウンロードするのは約150 MBです。

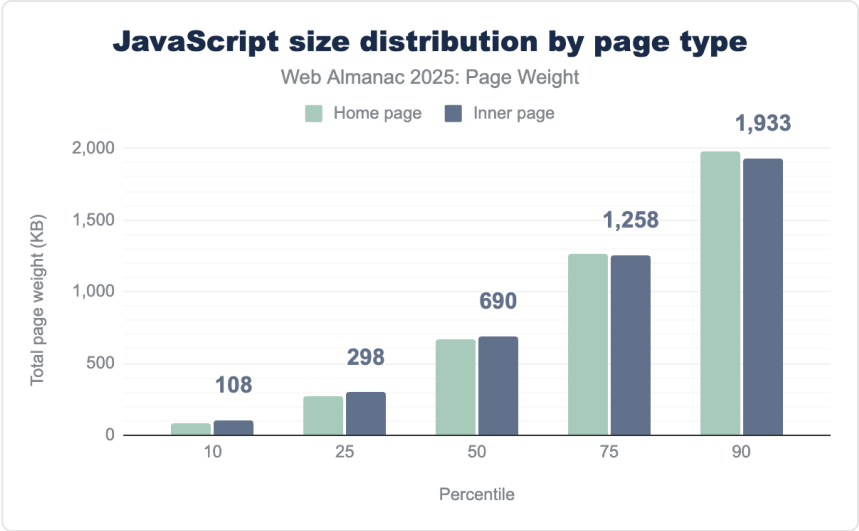


図14.19. ページタイプ別JavaScriptサイズの分布。

画像とは異なり、JavaScriptの使用はホームページと内部ページ間で一貫していました。第10パーセンタイルでは、ホームページは87 KB、内部ページは108 KBのJavaScriptを使用しました。ホームページの中央値は664 KBを使用し、内部ページの対応するページは690 KBを使用しました。第90パーセンタイルでは、ホームページと内部ページはそれぞれ1,979 KBと1,933 KBを使用しました。

未使用JavaScript

未使用のJavaScriptファイルは、無駄な機会だけでなくページの肥大化も表しています。呼び出されたすべてのスクリプトは、ページに貢献するかどうかに関わらず、ダウンロード、解析、コンパイル、実行されなければなりません。

このデータセットはWeb Almanacのクロールと同時に実行されたLighthouseテストからのものであることに注意することが重要です。未使用とは非圧縮後の未使用JavaScriptです。例えば、amazon.co.ukは423 KBのJavaScriptをダウンロードしますが、非圧縮すると1,721 KBになります。

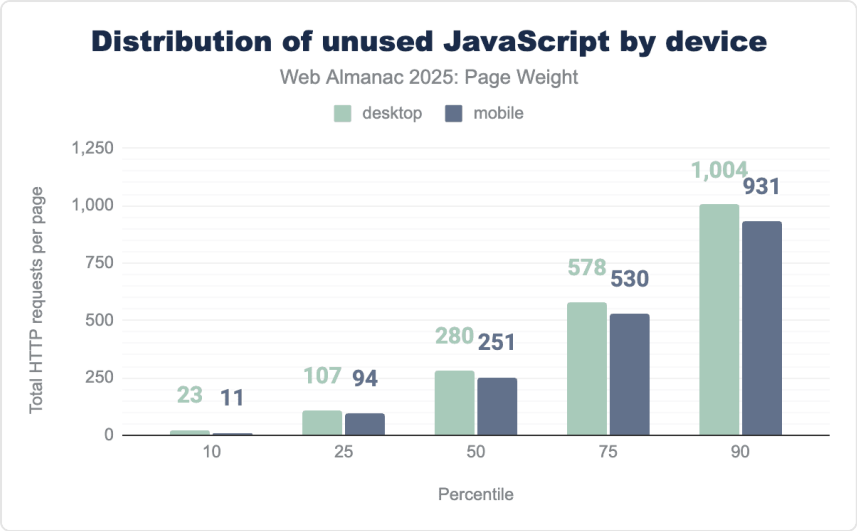


図14.20. デバイス別未使用JavaScriptの分布。

中央値ページでは、デスクトップで280 KB、モバイルで251 KBの非圧縮未使用JavaScriptが存在しました。第100パーセンタイルでは、デスクトップページは驚異的な203.2 MBの非圧縮未使用JavaScriptを示しました。

動画バイト

画像はウェブページで最も数多くのアセットであることが多いですが、動画バイトはデジタルペイロードの議論の余地のない重量級チャンピオンです。ウェブが「動画ファースト」の体験へとシフトするにつれ（バックグラウンドのヒーロー動画、ループする商品プレビュー、埋め込みクリップの活用など）、転送されるデータの量は前例のない水準に達しています。

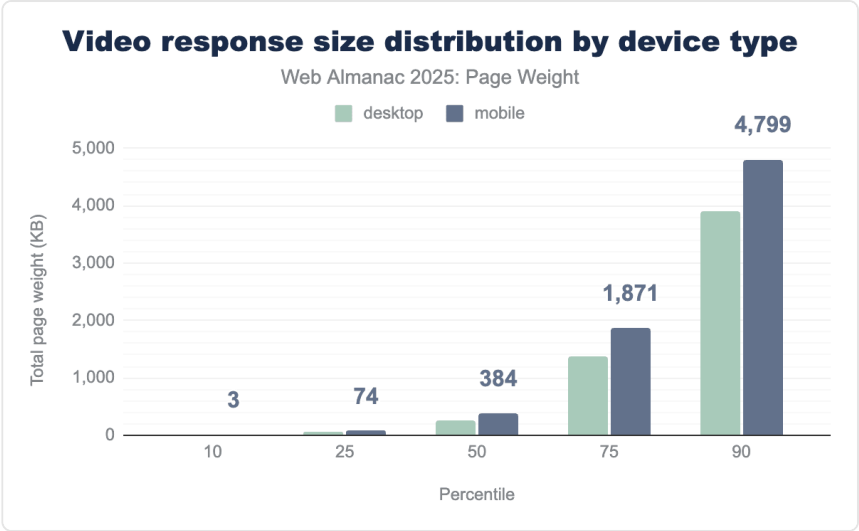


図14.21. デバイス別動画レスポンスサイズの分布。

中央値ページの動画バイトは、2024年の246 KBから2025年の315 KBへと前年比28%増加しました。モバイルはすべてのパーセンタイルで一貫してより多くの動画バイトを使用しました。第100パーセンタイルでは、ページは695 MBの動画を送信しました。文脈として、モバイルデバイスのユーザーがデータプランなしでこれらのページの1つを読み込もうとした場合（MBあたり0.15ドル[458]を支払うことになる）、動画バイトだけで104.10ドルのコストがかかる可能性があります。

458. *https://www.telus.com/en/mobility/prepaid/plans*

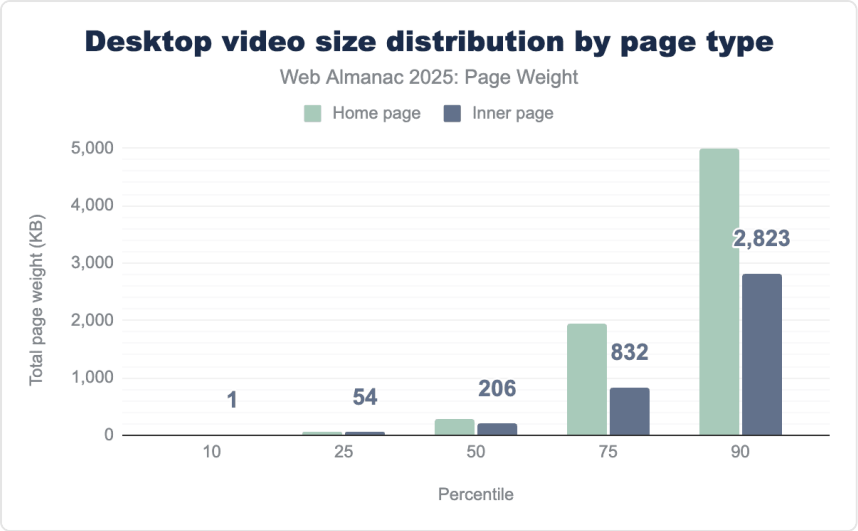


図14.22. ページタイプ別デスクトップ動画レスポンスサイズの分布。

デスクトップでは内部ページよりもホームページで多くの動画バイトが送信されました。動画を含むホームページの中央値は288 KBで、動画を含む内部ページは206 KBでした。ページタイプ間のギャップは第75パーセンタイルで最も大きく、ホームページは1,934 KBで、内部ページの832 KBより232.6%大きかった。

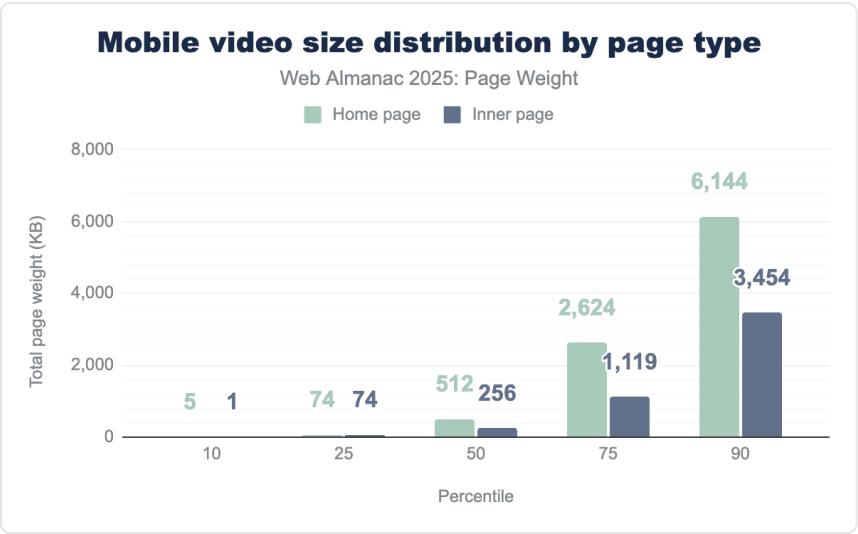


図14.23. ページタイプ別モバイル動画レスポンスサイズの分布。

ホームページでの動画バイトが多いパターンはモバイルデバイスでも続きました。ホームページの中央値は内部ページの256 KBと比較して2倍のバイト（512 KB）を送信しました。第90パーセンタイルでは、モバイルホームページは6.1 MBの動画を読み込みました。

いくつかの技術的な層が動画ファイルサイズに影響を与えます：

- **ファイルフォーマット：** 動画と音声ストリームを保持する「ラッパー」（例えばMP4、WebMなど）。コンテナ自体は軽量ですが、コンテナの選択は通常どの効率的なコーデックが使用できるかを決定します。
- **コーデック（圧縮/解凍）：** 動画を縮小するために使用される数学的アルゴリズムの効率性。H.264（AVC）のようなレガシーコーデックは広く互換性がありますが「重い」のに対し、HEVC（H.265）やAV1のような現代のコーデックは大幅に少ないバイト数でより優れた視覚的品質を提供します。
- **クロマサブサンプリング：** 輝度（ルーマ）情報よりも色情報の解像度を低くすることでファイルサイズを削減する方法で、人間の目が輝度により高い感受性を持つことを利用しています。

ファイルフォーマットは動画バイトに対する最も容易に観察できる要因です。WebMファイルはウェブ用に特別に設計されています。ブラウザの制約に合わせて構築されているため、WebMファイルはMP4より軽量であることが一般的に期待されています。

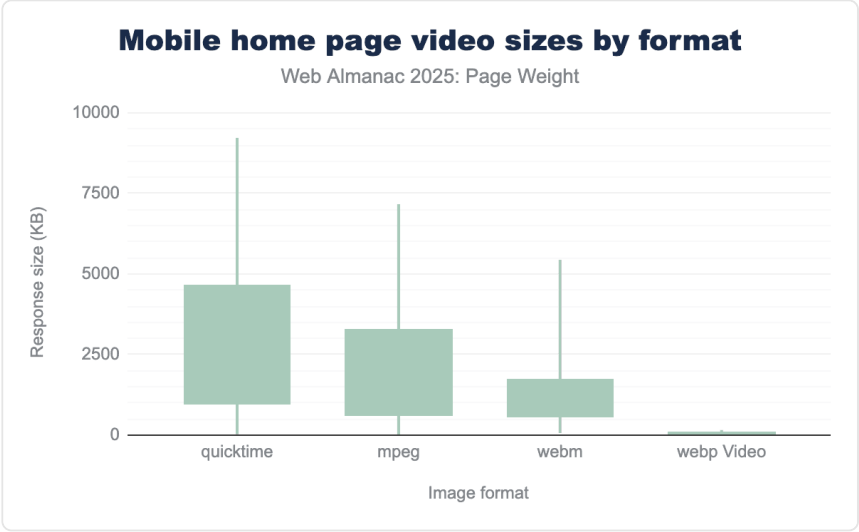


図14.24. フォーマット別モバイルホームページ動画サイズの分布。

モバイルホームページ動画の中央値パーセンタイルでは、QuickTimeファイルが最も高い中

央値レスポンスサイズ（976.0 KB）を持ち、次いでmpeg（639.5 KB）とWebM（581.8 KB）が続きます。WebP Videoは21.3 KBと大幅に小さい中央値サイズを持っています。WebP Videoファイルは最大サイズ5,743.5 KB、第90パーセンタイル155.6 KBと、測定されたすべてのパーセンタイルで一貫して最小です。

遭遇した最大のホームページデスクトップ動画ファイルはmpegで、第100パーセンタイルで326 MBでした。

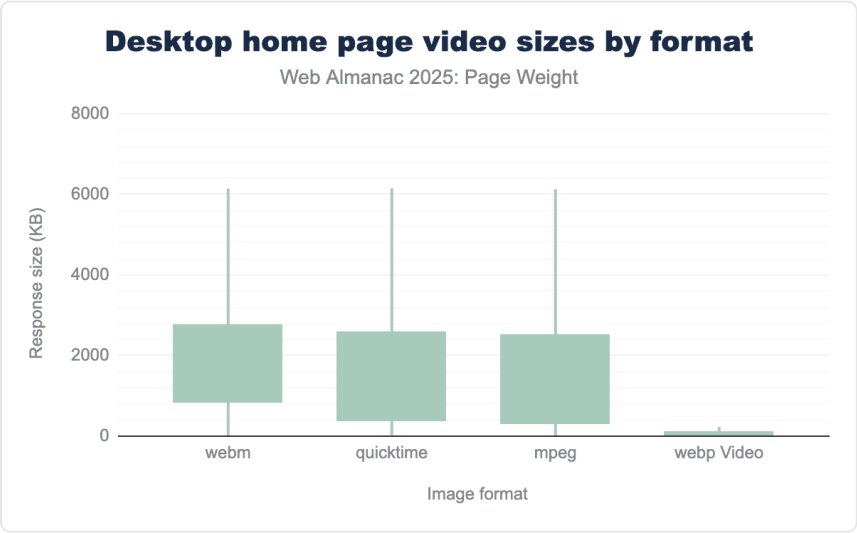


図14.25. フォーマット別デスクトップホームページ動画サイズの分布。

高端のサイズは類似していますが、中央値（第50パーセンタイル）のサイズはさまざまで、WebMが860.22と最も大きい中央値を持ち、次いでQuickTime（399.97 KB）とmpeg（326.52 KB）が続きます。これは、最大のファイルがほぼ同じサイズであるにもかかわらず、「典型的な」webmファイルがQuickTimeやmpegの対応ファイルより大きいことを示唆しています。

WebP Videoの中央値レスポンスサイズは32.58 KBで、これは他のフォーマットの中央値の10〜26倍小さいです。第90パーセンタイルでも、WebP Videoのサイズ228.33は非常に小さいです。これは、WebP Videoファイルの90%が他の3つのフォーマットの最小の90%よりも小さいことを意味し、他のフォーマットの第90パーセンタイルは6,100を超えています。

WebMの中央値サイズ（860.22）はQuickTime（399.97）の2倍以上、mpeg（326.52）のほぼ3倍で、すべてのフォーマットが非常に大きなファイルを処理できますが、WebMの分布の中心がより大きいサイズに偏っていることを示しています。

遭遇した最大のホームページデスクトップ動画ファイルはQuickTimeで、第100パーセンタ

イルで431 MBでした。

CSSバイト

カスケーディングスタイルシート（CSS）はウェブ上のページをスタイリングするために長年使用されており、レイアウトを作成してページの視覚的要素を制御する比較的軽量な方法です。

CSSはHTMLと並行して機能し、フォントスタイル、色、スペーシング、さらには要素の可視性を制御することで、より細かいレベルの制御を提供するとともに、ページの他の側面からの分離によって保守性を高めます。CSSは画像よりも軽量で、大きなアセットをCSSエフェクトやアニメーションに置き換えることでサイトパフォーマンスの向上に役立ちます。

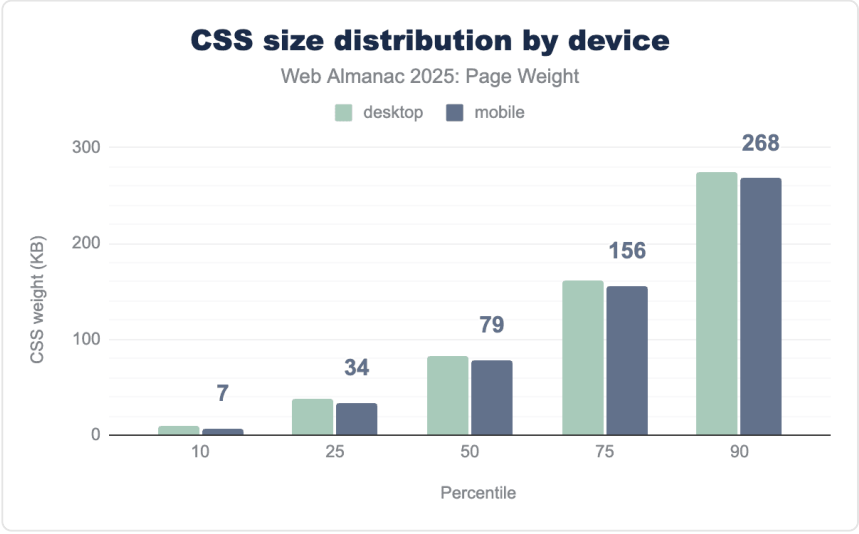


図14.26. デバイスタイプ別CSSレスポンスサイズの分布。

モバイルのCSSバイトは、昨年と比べて収縮が見られた下位10%を除くすべてのパーセンタイルで増加しました。

しかし、この成長はページの他の側面と比較するとほとんどが控えめです。第90パーセンタイルでは合計ファイルサイズはわずかに1/4 MBを超えるだけで、昨年から8 KBだけ増加しました。

第50パーセンタイルでは昨年から4 KBの増加で、これはCSSファイルに追加された約4,000文字に相当し、継続的な多用を示しています。

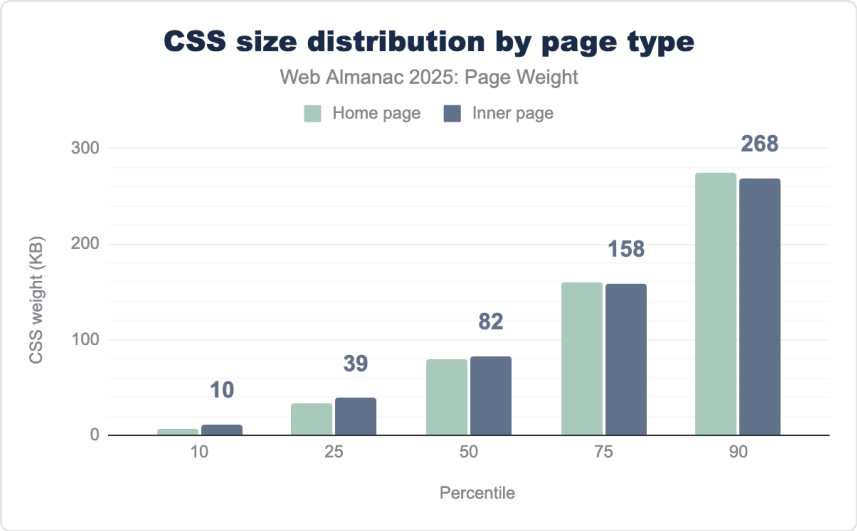


図14.27. ページタイプ別CSSレスポンスサイズの分布。

内部ページは、再び1 KBの収縮が見られた最低10%を除くすべてのパーセンタイルで成長するという同じトレンドに従っています。 内部ページは第25、50、75パーセンタイルでCSSの使用率がわずかに高い特徴があります。

特に、結果の上位端、第75と第90パーセンタイルに達すると、ホームページが実際にはより大きなCSSサイズを持っています。これはCSSの非効率な使用、つまり現在のページでの使用に関わらず大部分のファイルを読み込んでいること、またはすべてのページのビジュアルを提供するためにCSSへの依存が重くなっていることによる可能性があります。

HTMLバイト

HTMLバイトとは、ページ上のすべてのマークアップの純粋なテキストの重さを指します。通常は、`<div>`や`<span>`のようなドキュメント定義と一般的に使用されるページタグを含みます。しかし、スクリプトタグの内容や他のタグに追加されたスタイリングなどのインライン要素も含まれています。これはHTMLドキュメントの肥大化を急速に引き起こす可能性があります。

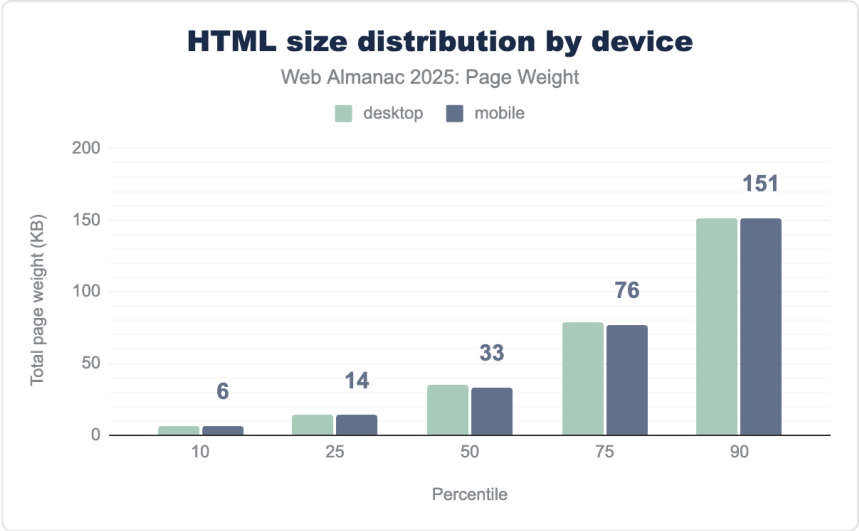


図14.28. デバイスタイプ別HTMLレスポンスサイズの分布。

HTMLサイズは第10と第25パーセンタイルでデバイスタイプ間で均一でした。第50パーセンタイルから、デスクトップのHTMLがわずかに大きくなりました。デスクトップが401.6 MB、モバイルが389.2 MBに達した第100パーセンタイルまで意味のある差は現れませんでした。

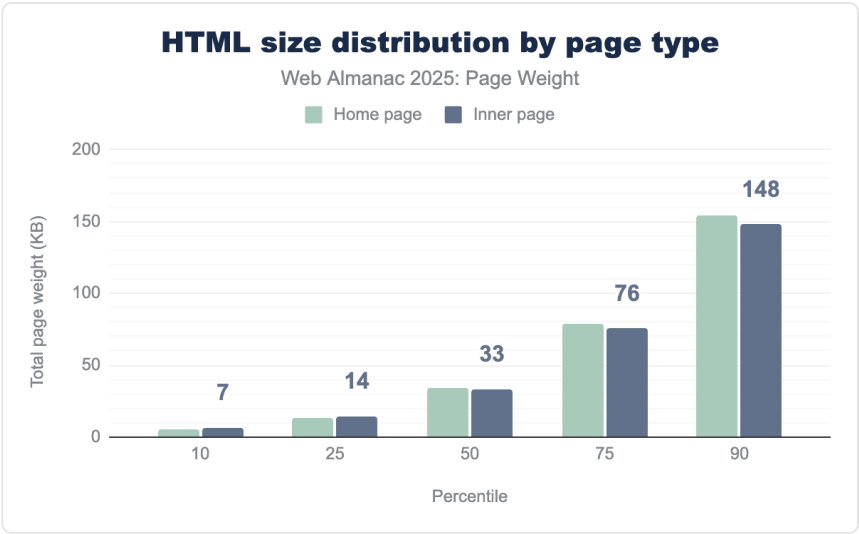


図14.29. ページタイプ別HTMLレスポンスサイズの分布。

HTMLサイズでは内部ページとホームページ間の差はほとんどなく、第75パーセンタイル以上でようやく明らかになります。第100パーセンタイルでは、差は顕著です。内部ページのHTMLは驚異的な624.4 MBに達し、ホームページのHTML 166.5 MBより375%大きいです。

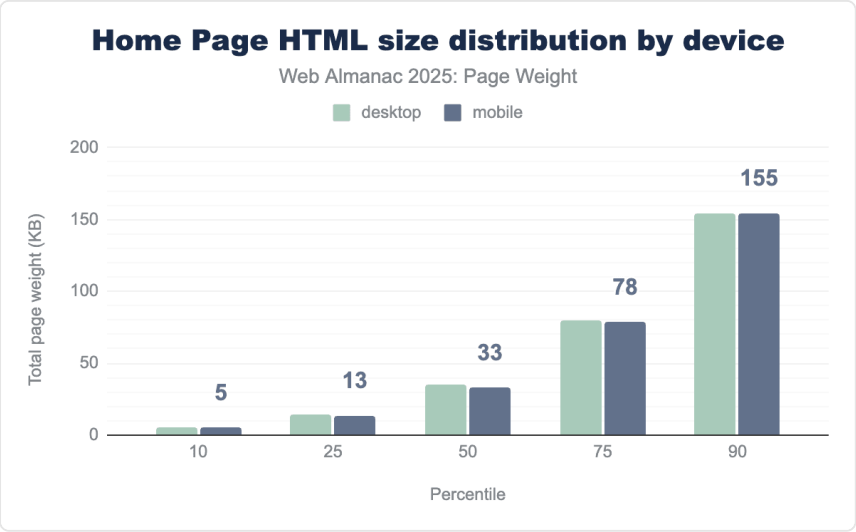


図14.30. デバイスタイプ別ホームページHTMLレスポンスサイズの分布。

モバイルとデスクトップのサイズの違いは極めて軽微で、これはほとんどのウェブサイトがモバイルとデスクトップユーザーの両方に同じページを提供していることを示唆しています。

このアプローチは開発者のメンテナンス量を劇的に削減しますが、実質的にサイトの2つのバージョンが1つのページに展開されるため、全体的なページ重量が高くなる可能性があります。

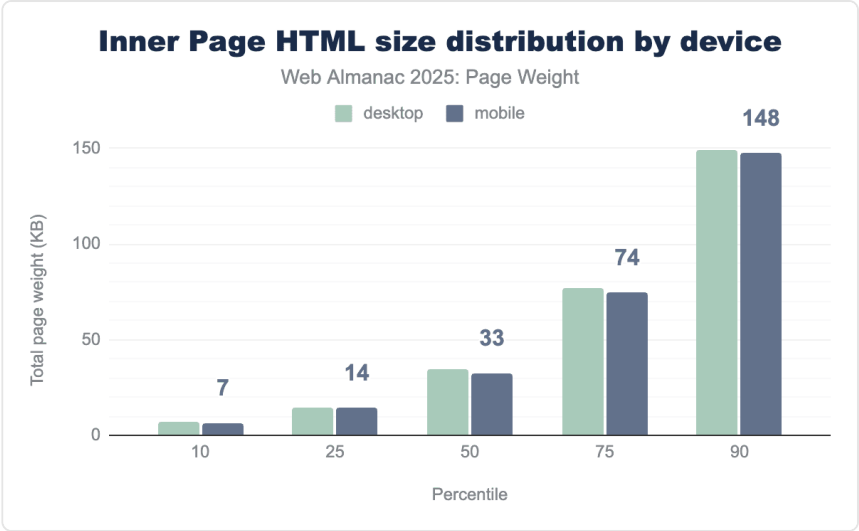


図14.31. デバイスタイプ別内部ページHTMLレスポンスサイズの分布。

内部ページは第90パーセンタイルまで、ホームページの対応するものと同様の動作をしました。第100パーセンタイルでは、最大の内部ページHTMLレスポンスは両デバイスタイプで同じ624.4 MBでした。標準的な10〜25 Mbpsの接続では、これの読み込みに30〜90秒かかります。

フォントバイト

画像と動画がページ肥大化の最も明白な要因である一方、ウェブフォントは現代のページ重量の微妙だが重大な部分を占めています。設計フェーズで見落とされることが多い、システム標準フォントからカスタム「ウェブフォント」への移行は、新たなレイテンシの層をもたらしました。タイプフェイスのすべてのユニークなウェイト（太字、ライト、シン）とスタイル（イタリック）は個別のファイルダウンロードを必要とし、単一のフォントファミリーが数百キロバイトの追加データへと急速に膨らむ可能性があります。

これらのフォントの「重さ」は3つの主要な要因によって決まります：

1. **文字サポート：** 複数の言語（ラテン語、キリル文字、ギリシャ語）をサポートするフォントには、基本的な英語のみのフォントよりも何千もの「グリフ」（文字の形）が含まれています。
2. **スタイルとウェイト：** 各バリエーション（例えば、Roboto Regular、Roboto Bold、Roboto Bold Italic）は通常、個別のファイルです。完全な「ファミリー」を読み込むことは、大きな最適化された画像よりも重くなる可能性があります。
3. **フォーマット効率：** TTF（TrueType）やOTF（OpenType）のような古いフォー

マットは非圧縮ですが、WOFF2のような現代のウェブ専用フォーマットはBrotli圧縮を使用して品質を失わずにファイルサイズを最大30%削減します。

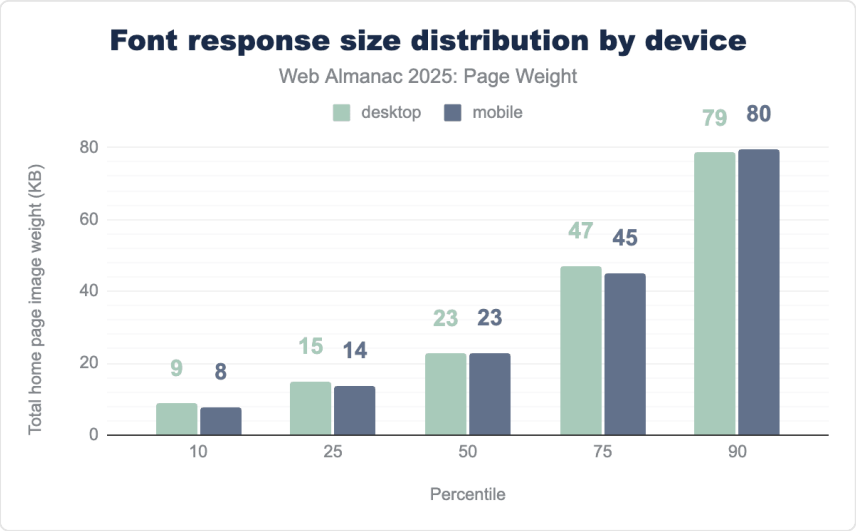


図14.32. デバイスタイプ別フォントレスポンスサイズの分布。

中央値ページは23 KBのフォントを送信しました。送信バイトは第90パーセンタイルまでデバイス間で緊密に一致し続けました。ホームページと内部ページは中央値を23 KBとして、パーセンタイル全体にわたってまったく同じバイト量を送信しました。詳細については、フォントチャプターを参照してください。

その他のファイルタイプ

最終的にウェブは多くのファイルタイプを持つ多様なエコシステムです。あまり見かけないファイルタイプには、中央値ページとデバイスタイプ全体で15 KBのWASM、12 KBの音声が含まれます。JSON、XML、テキストファイルタイプは中央値ページで0 KBの使用でした。少なくとも1つのサイトがモバイルページで117.6 MBのJSONを送信しており、最大の低頻度ファイルタイプのタイトルを獲得していることは注目に値します。

リクエスト量で見るページ重量

バイトの総数がデータの生の量を測定するのに対し、リクエストの量はブラウザとサーバー間の会話の複雑さを測定します。両方のレンズを通してページの重さを研究することは不可欠です。なぜなら「軽い」ページ（バイト数で）でも、レンダリングに多すぎる個別のリク

エストを必要とする場合は「遅い」場合があるからです。バイトはページをダウンロードするために必要な帯域幅を表します。リクエストの量はウェブアーキテクチャに固有のレイテンシのオーバーヘッドを表します。

中央値ページのファイルタイプ量

ページの重さに最も貢献しているファイルタイプを理解するために、中央値ページ（第50パーセンタイル）のファイルタイプリクエストを調べました。このアプローチは、各ファイルタイプの全体的な影響を測定するためのベースラインを確立します。

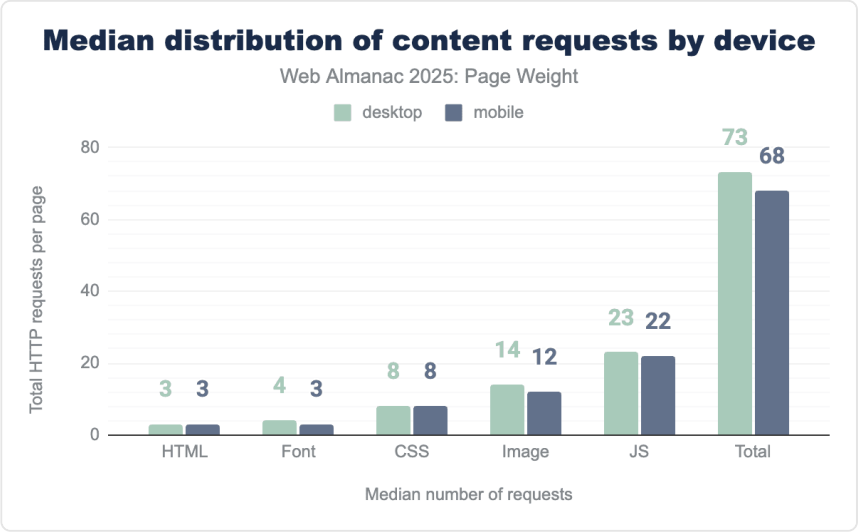


図14.33. コンテンツタイプとデバイスタイプ別のリクエスト数の中央値。

中央値ページはデスクトップで77リクエスト、モバイルで72リクエストを行います。全体的なリクエストはモバイルで前年比9%、デスクトップで8%増加しました。画像ファイルタイプは2025年も下降トレンドを続け、前年比6%減少しました。

ファイルタイプ別リクエストは前年から概ね一貫していました。デスクトップの中央値は77のリソースを呼び出し、モバイルの対応するものは72だけを必要とします。返されるHTMLファイルの数は両方のページタイプで2から3に増加しました。ホームページの中央値のCSSは、前年から変化なしで安定した内部ページの使用と一致するようにわずかに増加しました。JavaScriptは最も多くリクエストされるファイルタイプであり続けました。

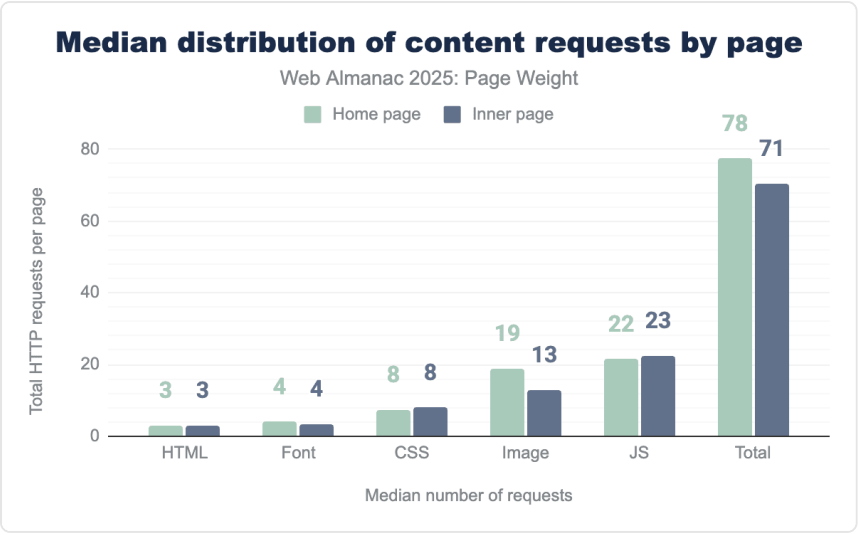


図14.34. コンテンツタイプとページタイプ別のリクエスト数の中央値。

ホームページの中央値は3つのHTMLファイル、4つのフォントファイル、8つのCSSファイル、19の画像、22のJavaScriptファイル、合計78ファイルを読み込みます。内部ページの中央値は3つのHTMLファイル、4つのフォントファイル、8つのCSSファイル、13の画像、23のJavaScriptファイル、合計71ファイルを読み込みます。

リクエスト量分布

中央値の動作が確立されたので、パーセンタイル全体のデータを見ることができます。パーセンタイルはデータポイントがどのように分散しているかを明確に示し、パターンの解釈を容易にします。

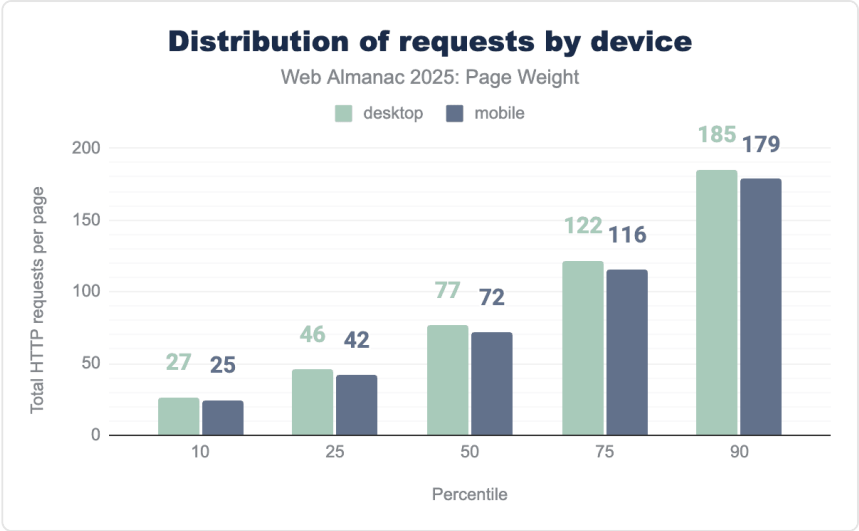


図14.35. デバイスタイプ別リクエスト量の分布。

最も軽いページでも今年は重くなりました。第10パーセンタイルでは、最もシンプルなページが今年3つのリクエストを追加しました。対照的に、第90パーセンタイルでは9つの追加ファイルがリクエストされました。これは、極端な例で示されるように、モバイルとデスクトップが同じ数のリクエストを行うことを示しています。

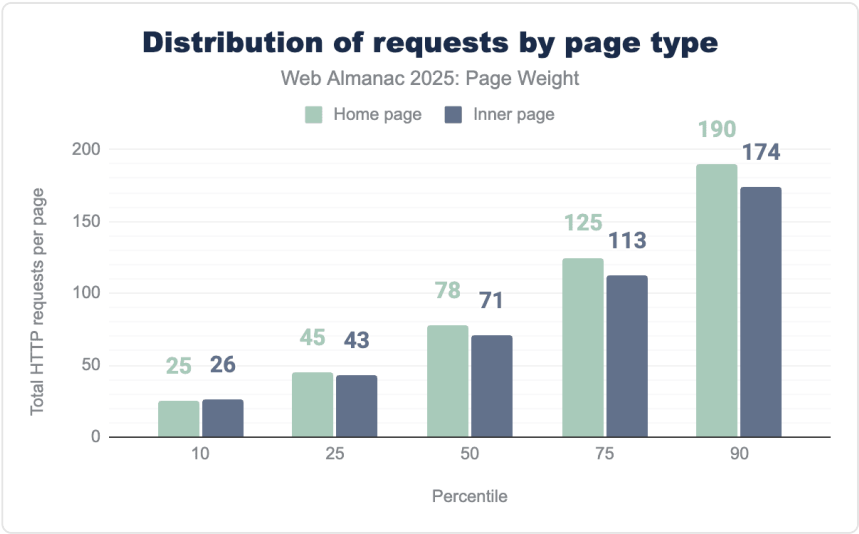


図14.36. ページタイプ別リクエスト量の分布。

ホームページと内部ページでグループ化したデータを見ると、ファイルリクエストの増加はすべてのパーセンタイルで一貫していました。中央値ページはデスクトップで77リクエスト、モバイルで72リクエストを行います。ホームページと内部ページの差はパーセンタイルが上がるにつれて広がります。

これは特に興味深い点で、モバイルとデスクトップでデータを見るとリクエスト数に一貫性が見られ、ホームページのリクエストが重要な差別化要因であることを示しています。

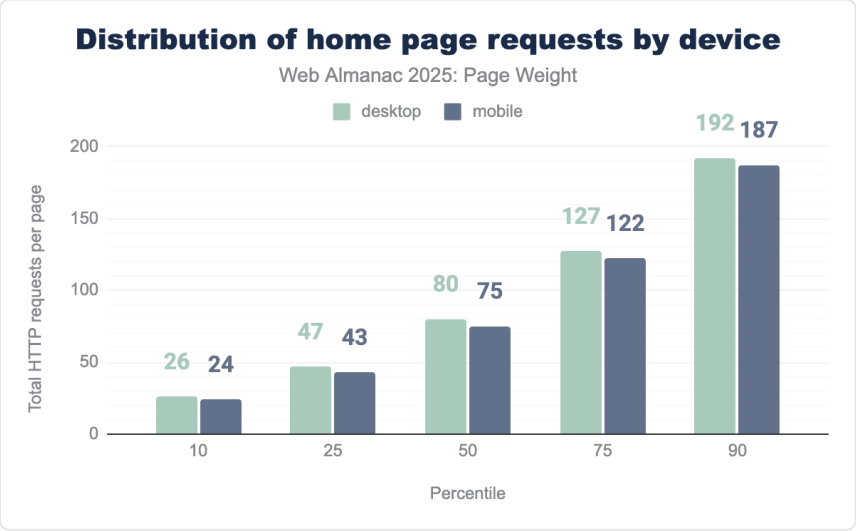


図14.37. デバイスタイプ別ホームページリクエスト量の分布。

デバイスタイプ別のホームページリクエストを見ることで詳細な分析が可能です。各デバイスはホームページと内部ページのデータが混在した場合よりもわずかに多くのリクエストを必要とします。デスクトップホームページの中央値は80リクエストを行い、モバイルの対応するものは75を行います。デバイス間で動作は一貫しており、パーセンタイルにわずかな差異があります。

画像リクエスト量

インターネットは猫の写真のために作られたという人もいます。これは献身的な動物愛好家からの誇張かもしれませんが、視覚的コンテンツがウェブを支配しているという事実を浮き彫りにしています。

画像は商品の確認から信頼性の高いニュース報道まで、あらゆる面で重要な役割を果たしています。しかし、これらの静的ファイルはしばしば最も重いデジタルリソースの一つであり、パフォーマンス最適化と技術革新の主要な候補となっています。

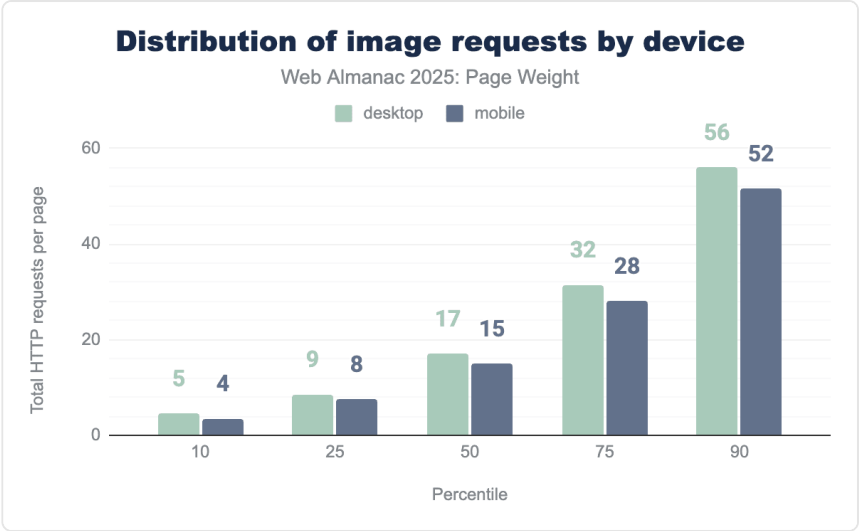


図14.38. デバイスタイプ別画像リクエストの分布。

第50パーセンタイルでは、デスクトップは17、モバイルは15の画像リクエストを行いました。デスクトップはパーセンタイル全体で一貫してより多くの画像ファイルを使用しました。これはデスクトップページが18枚の画像を呼び出し、モバイルが16枚をリクエストした2024年からわずかに減少しています。第100パーセンタイルでは、デスクトップページは26,330枚の画像をリクエストしました。そのページがどれほど素晴らしいものかは想像するしかありません。

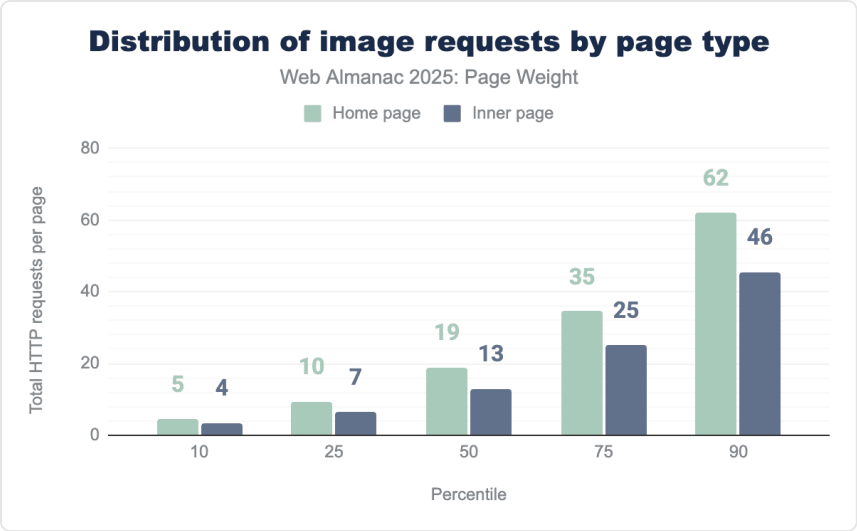


図14.39. ページタイプ別画像リクエストの分布。

ホームページと内部ページを比較すると、差は一貫しています。ホームページの中央値は19枚の画像を読み込み、内部ページでは13枚です。このパターンは全分布範囲にわたって維持され、ホームページが平均20の画像リクエスト、内部ページが14だった2024年のデータと非常に近いです。

画像リクエストにはわずかな減少があり、これは第75と第90パーセンタイルでも見られるトレンドですが、最小限の変化はサイト所有者やパブリッシャーの画像へのアプローチに大きな転換がないことを示唆しています。代わりに、ウェブの画像使用はいつも通りのビジネスとして現れています。

CSSリクエスト量

カスケーディングスタイルシート（CSS[459]）はウェブのプレゼンテーション層として機能します。マークアップ言語で書かれた文書のレイアウトを視覚的に表現することを開発者が定義できるスタイル言語です。Web Almanacの範囲では、HTMLに限定されます。場合によってはSVGも含まれることがあります。

CSSにより開発者は色、フォント、レイアウト、サイズなどの要素を指定でき、HTMLで定義された構造的コンテンツからプレゼンテーションを分離します。レスポンシブデザイン、アニメーションのタッチ、洗練されたレイアウトのための興味深い機能を提供するなど、ますます強力になり続けています。かつてJavaScriptを必要とした視覚的効果は今やCSSで実現

459. https://web.dev/css

でき、CSSは現代のウェブデザインの基盤となっています。

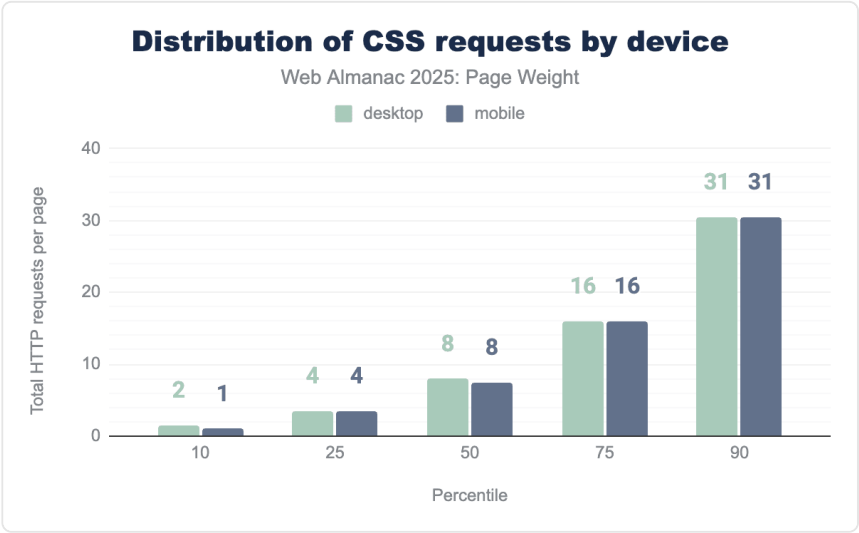


図14.40. デバイスタイプ別CSSファイルリクエストの分布。

デスクトップとモバイルデバイスの両方にわたって、ページがリクエストするCSSの量にはほとんど差がありませんでした。第50パーセンタイルでは、モバイルとデスクトップの両方が8つのCSSリソースをリクエストし、全分布にわたって1リソースリクエスト以上の差はありませんでした。これは2024年のパターンと非常に一致しており、前年比でのリクエスト数は第90パーセンタイルまでおおむね安定していました。

第90パーセンタイルでは、今年はデスクトップで31、モバイルで30のCSSリクエストが行われ、デスクトップとモバイルの両方で26だった2024年よりわずかに増加しました。これは、開発者がどれだけの別々のアセットに分割しているか、またはどのようなツールを使用しているかという点で、過去12ヶ月間でCSSへのアプローチにほとんど変化がないことを示唆しています。

また、開発者がクライアントタイプに関わらず同じCSSファイルを提供する可能性がはるかに高いことも示しています。

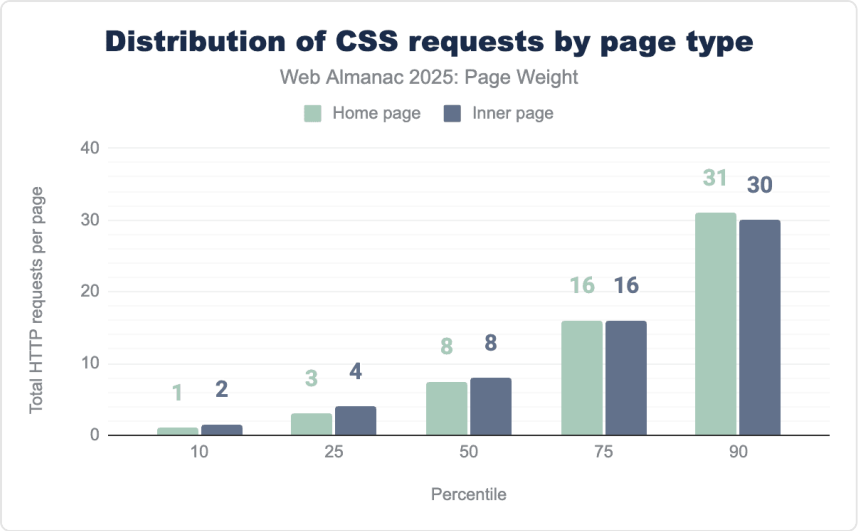


図14.41. ページタイプ別CSSファイルリクエストの分布。

中央値ページは8つのCSSリクエストを行いました。内部ページは第75パーセンタイルまでホームページよりも多くのリクエストを行いました。第90パーセンタイルでは、ホームページが内部ページより1つ多くのリクエストを行いました。

ホームページと内部ページでリクエストされたファイルの数にはほとんど変化がありません。この分析から同じファイルが提供されているかどうかは判断できませんが、開発者がサイト全体にわたって1セットのCSSリソースを提供しているという事実を示しており、これを分割して実際のページに必要なCSSだけを送ることで個々のページのCSSの重さを削減する機会を逃している可能性があります。

JavaScriptリクエスト量

画像と動画が静的な重さである一方、JavaScriptは能動的な重さです。その影響は独特で、リソースを2回消費します。ダウンロード時（ネットワーク）と実行時（CPU）です。JavaScriptが多数のリクエストに分割されると（モジュラーコーディングのプラクティスやサードパーティプラグインによることが多い）、パフォーマンスに「二重の税金」をもたらします。

合計バイト数が少ない場合でも、スクリプトリクエストの量が多いとブラウザのメインスレッドが麻痺し、読み込まれているように見えるがユーザー入力に反応しないサイトになる可能性があります。

98.1%

図14.42. JavaScriptを使用しているページ。

すべてのページの98.1%が少なくとも1つのJavaScriptファイルのリクエストを行っています。この数字にはHTMLに含まれているインラインJavaScriptは含まれていません。

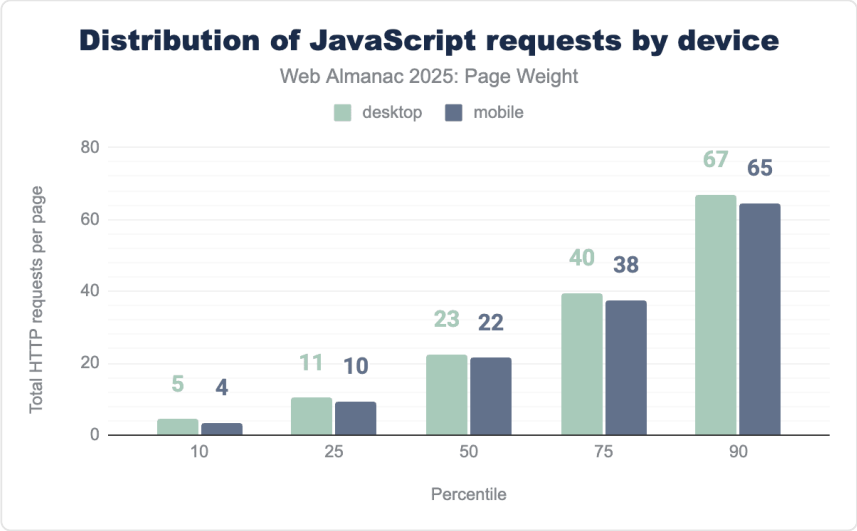


図14.43. デバイスタイプ別JavaScriptリクエストの分布。

デスクトップの中央値ページはデスクトップで23、モバイルで22のJavaScriptリクエストを行いました。デスクトップページは一貫してモバイルよりも多くのJavaScriptファイルを要求しました。デスクトップホームページは第100パーセンタイルで12,676ファイルがリクエストされ、最も多くのJavaScriptファイルを要求しました。Web Almanacクローラーは今年、年間の冬眠に値します。

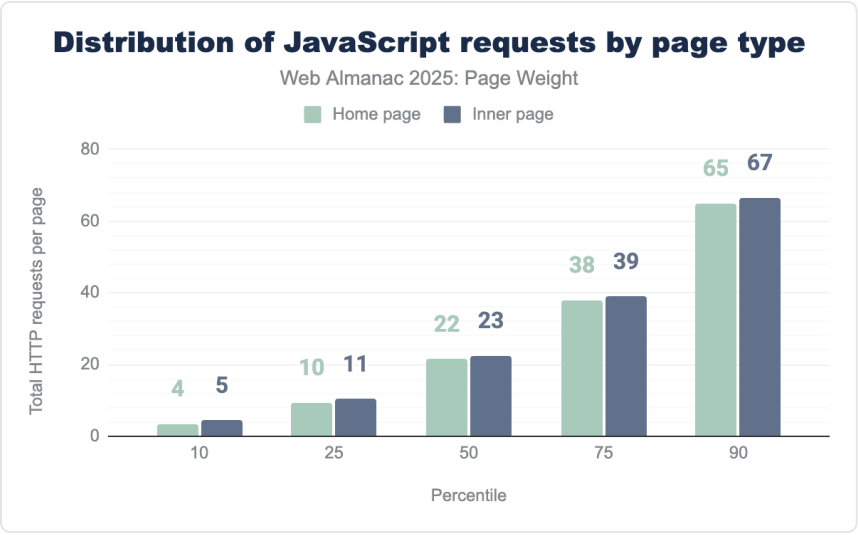


図14.44. ページタイプ別JavaScriptリクエスト量の分布。

内部ページは一貫してホームページよりわずかに多くのJavaScriptファイルを要求しました。ホームページの中央値は22のJavaScriptファイルをリクエストし、内部ページは23をリクエストしました。

フォントリクエスト量

フォントファイルはウェブサイトがウェブサイトのテキスト部分のスタイリングを指定することを可能にします。通常、大文字と小文字のA〜Zのすべての文字で構成されています。これにより、ブラウザはシステムにインストールされたフォント以外のスタイルにテキストを変換できます。

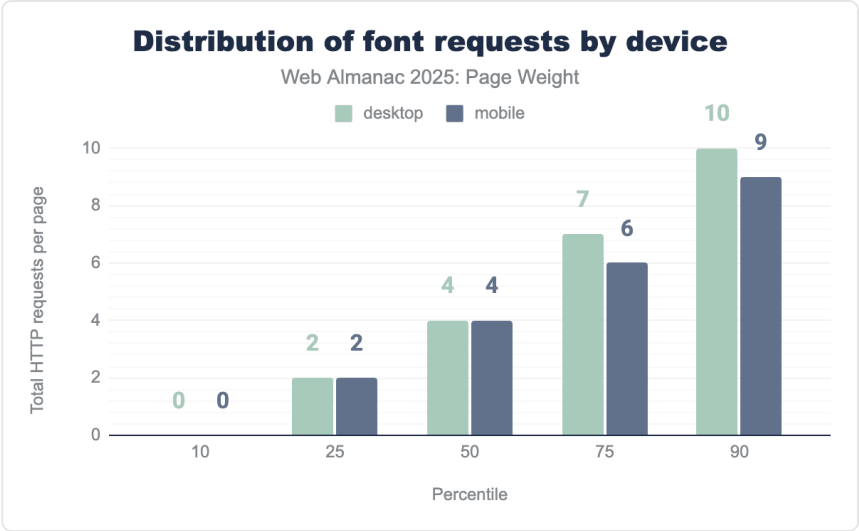


図14.45. デバイスタイプ別フォントリクエストの分布。

下位10パーセンタイルは、モバイルとデスクトップの両方でシステムにインストールされたフォントに代わりに依存し、追加のフォントリクエストを使用しません。モバイルのフォントリクエストは第75パーセンタイル以上では平均1つ少ない状態で低くなっています。

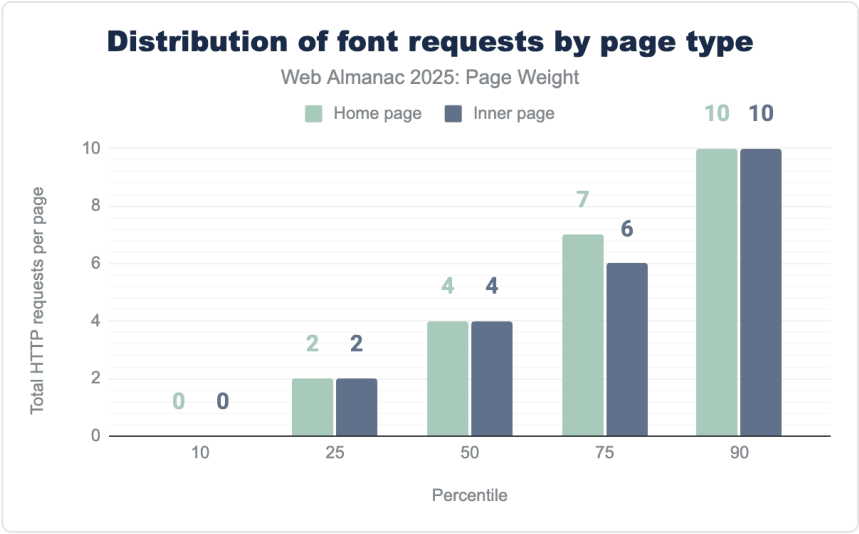


図14.46. ページタイプ別フォントリクエストの分布。

ホームページと内部ページは第75パーセンタイルを除いて平均リクエスト数が同一です。デスクトップホームページは第100パーセンタイルで3,038ファイルがリクエストされ、最も多くのフォントを読み込みました。問題のサイトがQ1でパフォーマンス最適化を優先したフォントリポジトリであることを本当に願っています。

一般的に、個々のページで異なるフォントが使用されるのではなく、サイト全体のスタイルとして使用されることが期待されます。プロモーション素材のような例外はありますが、平均はインポートされたフォントのほとんどがサイト全体にわたって使用されていることを示しています。

その他のリクエスト量

HTMLフラグメント、JSONデータ、サードパーティアセットを含む「ユーティリティ」ファイルのリクエスト量は、大きな累積的な「レイテンシ税」を生み出します。特にマイクロサービスとモジュラーデザインの台頭により、現代のウェブアーキテクチャでは、単一のページ読み込みが小さく分散したファイルへの数十のリクエストをトリガーする可能性があります。これらのファイルがバイト単位で軽量であっても、リクエストの量だけでブラウザのネットワークスタックを圧倒し、ページのレンダリングを遅延させる可能性があります。

中央値ページはHTMLで2リクエスト、その他のファイルタイプで2リクエストを行いました。第90パーセンタイルでは、これはHTMLで12、その他のファイルタイプで12に上昇します。それでも、害のないファイルタイプの新記録を打ち立てることに専念している人もいます。どこかに、ページ読み込みで17,065のその他のファイルタイプをリクエストするデスクトップ内部ページがあります。彼らは第100パーセンタイルであるという夢を達成しました。よくやった、相棒。JSONを置いてください。

バイト節約技術の採用率

ページ重量を削減する最もシンプルな方法は、そもそもバイトを送信しないことです。ウェブページの構築方法に気を配り、リソースの最適化を作業の標準的な部分にすることは、特定の技術を検討する前でも、ページの全体的な重量に大きな影響を与えることができます。

これに加えて、ネットワーク上で送信されるデータ量を削減するために採用できる特定の技術とテクニックもあります。このセクションでは、サードパーティ埋め込みのファサード、最小化、圧縮の3つを見ていきます。

動画やその他の埋め込み用のファサード

ファサード（インタラクション時のインポートとも呼ばれる）は、動画埋め込み、ソーシャルメディアの投稿、またはチャットウィジェットなどの重いアイテムをシンプルなグラフィ

ック表現に置き換えるデザインパターンで、ユーザーがインタラクションする際にのみ完全なインタラクティブな要素に置き換えられます。

ユーザーがクリックするとYouTube埋め込みと関連するすべてのデータとリソースを読み込む、動画のシンプルなサムネイルを想像してください。

これはユーザーがページ読み込み時にこれらの追加リソースをダウンロードしようとしないため、パフォーマンスの向上をもたらし、最終的にページの重量を節約できます。ユーザーがその動画を見たくない、またはチャットボットセッションを開きたくない場合、それらのバイトはリクエストされません。

ファサードを使用しているサイトの検出は実際にはほとんど不可能です。しかし、Lighthouseはバージョン13のリリース前に、ファサードを使用したサードパーティリソースの遅延読み込み[460]と呼ばれる、認識されているファサードソリューションを持つ一般的な埋め込みのテストを持っていました。今年のクロールはバージョン12で実行されたため、このメトリックにアクセスできました。

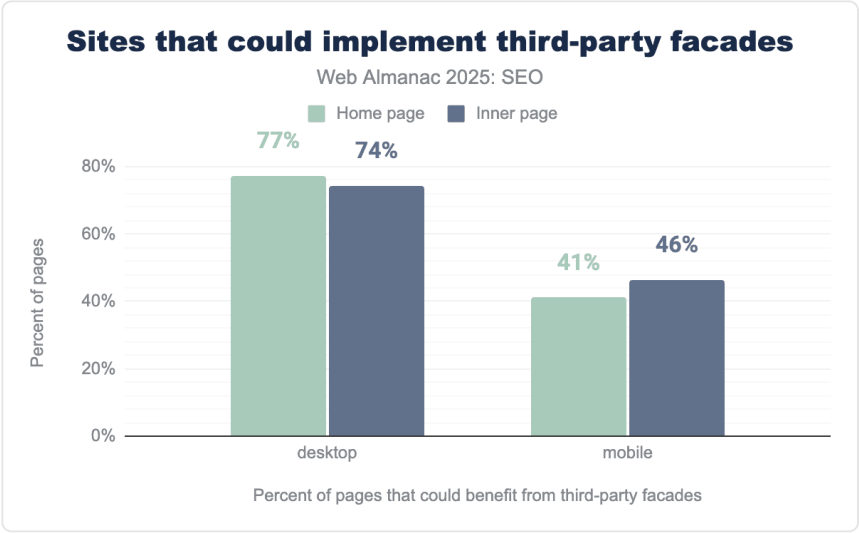


図14.47. サードパーティファサードを実装できるサイト。

モバイル読み込みのホームページの41%とデスクトップクライアントで読み込まれたホームページの77%がファサードを使った遅延読み込みを実装する機会があると検出されました。これは、70%が検出されたデスクトップの2024年からの後退を表しますが、モバイル読み込みは45%からの改善を示しています。

460. https://developer.chrome.com/docs/lighthouse/performance/third-party-facades

2024年と同様に、モバイルページはデスクトップページよりもファサードを実装する可能性が高いようです。

興味深いことに、内部ページはデスクトップで74%と高いスコアを示した一方、モバイルでは46%と低かった。おそらく開発者はモバイル訪問者のホームページパフォーマンスに多くの焦点を当てているのでしょうか？

数字はページ重量を削減する大きな機会があることを示していますが、同様に独自の問題もあります。ユーザーが要素とインタラクションしてから知覚できる遅延や、特に動画では再生を開始するためにダブルクリックまたはタップが必要な場合があります。

しかし、ページ上の埋め込みを評価し、このアプローチの実装から恩恵を受けられるものがないか確認することは価値があります。

圧縮

圧縮とは、リソースをネットワーク経由でデバイスに送信する前にサイズを削減し、その後デバイスで解凍するプロセスです。このプロセスは通常、特にネットワークが最大のボトルネックである場合に、ページの読み込みを高速化します。

HTML、CSS、JavaScript、JSON、SVG、ico、ttfフォントファイルのようなテキストベースのファイルへのHTTP圧縮は、ネットワーク上でのページ重量を大幅に節約できます。メディアのような他のファイルタイプには恩恵が及ばないことに注意する価値があります。これらのファイルではエンコーディングの一部として圧縮が行われています。

GZip[461]、Brotli[462]、または圧縮の新参者であるZstandard (zstd)[463]を使用してHTTPリクエストを圧縮することで、これらのテキストベースリソースの重さを大幅に削減し、パフォーマンスを向上させることができます。

圧縮がページ重量の問題を完全に解消するわけではないことに注意する必要があります。フルサイズに解凍する必要があり、圧縮と解凍のプロセスは、過度に行われると全体的な速度を低下させる可能性がありますが、それはまれなことで、確かにネットワーク上で節約されるバイト数より優れています。

461. *https://developer.mozilla.org/docs/Glossary/gzip_compression*
462. *https://developer.mozilla.org/docs/Glossary/Brotli_compression*
463. *https://developer.mozilla.org/docs/Glossary/Zstandard_compression*

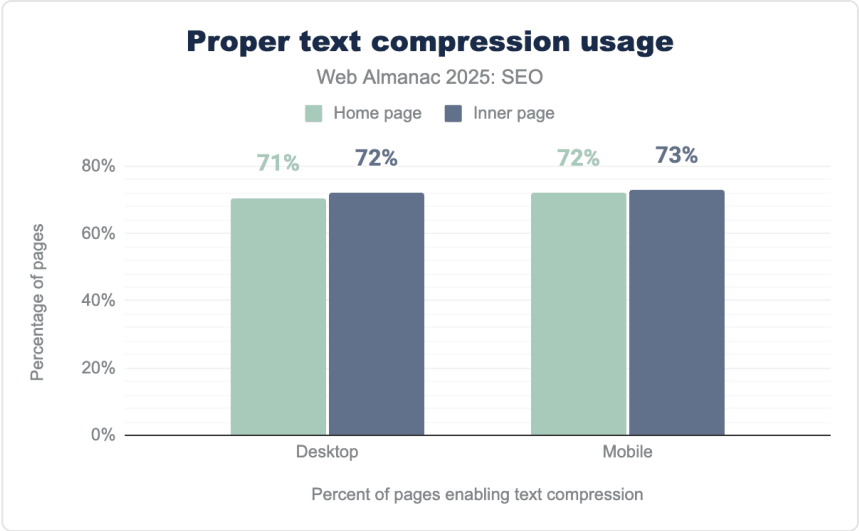


図14.48. 適切なテキスト圧縮の使用状況。

2025年、LighthouseのEnable text compression監査[464]で測定したところ、デスクトップホームページ読み込みの71%、モバイルホームページ読み込みの72%が適切なテキスト圧縮を使用していました。

内部ページでは、通過率はわずかに高く、デスクトップで72%、モバイルで73%でした。

これは、ホームページのデスクトップ通過率が70%、モバイルが71%で、内部ページがデスクトップ71%、内部ページ72%に達した2024年と比較して小さな改善を示しています。言い換えれば、各セグメントが前年比でわずか1パーセントポイント改善されました。

どんな増加もポジティブですが、すべてのデバイスとページタイプにわたって25%以上のページが効率的な圧縮を使用できていないことは、まだやや残念です。

最小化

テキストリソースの最小化[465]は、スペース、コメントなどの不要なデータを削除し、関数名や変数名を短くしてより小さなファイルを残すこともできます。

最小化にはクライアントサイドの追加オーバーヘッドはありません。つまり、ファイルは動作するために最小化解除される必要はありません。オンザフライで行う場合、オーバーヘッドは純粋にサーバーサイドです。

464. https://developer.chrome.com/docs/lighthouse/performance/uses-text-compression/
465. https://developer.mozilla.org/docs/Glossary/Minification

しかし、CSSやJavaScriptのような多くのリソースでは、最小化はビルドプロセスの一部として事前に処理できます。一度だけ行う必要があります。

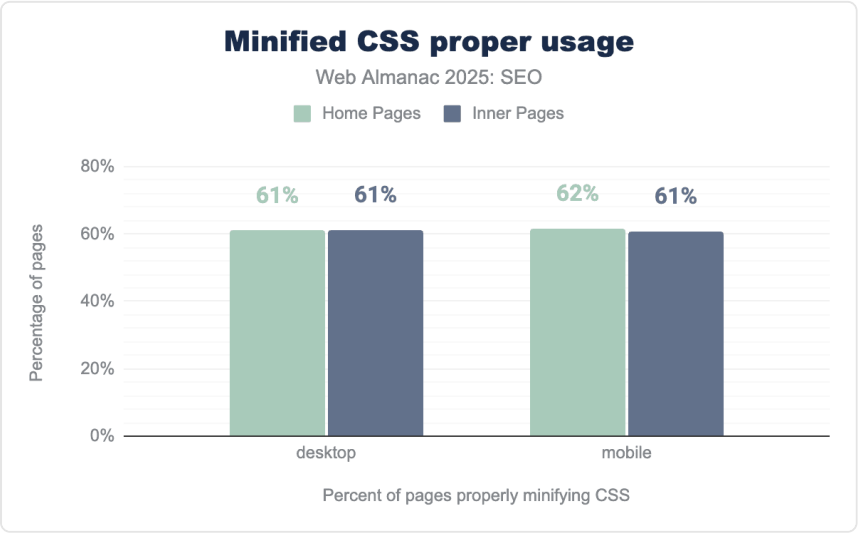


図14.49. CSSの最小化の適切な使用状況。

2025年の分析では、デスクトップの61%、モバイルホームページ読み込みの62%、デスクトップとモバイルの内部ページ読み込みの61%がLighthouseのCSS最小化[466]監査を通過しました。

これは2024年からわずかな低下を意味し、各メトリックが1%低下するという小さいながらも残念なトレンドです。

466. *https://developer.chrome.com/docs/lighthouse/performance/unminified-css/*

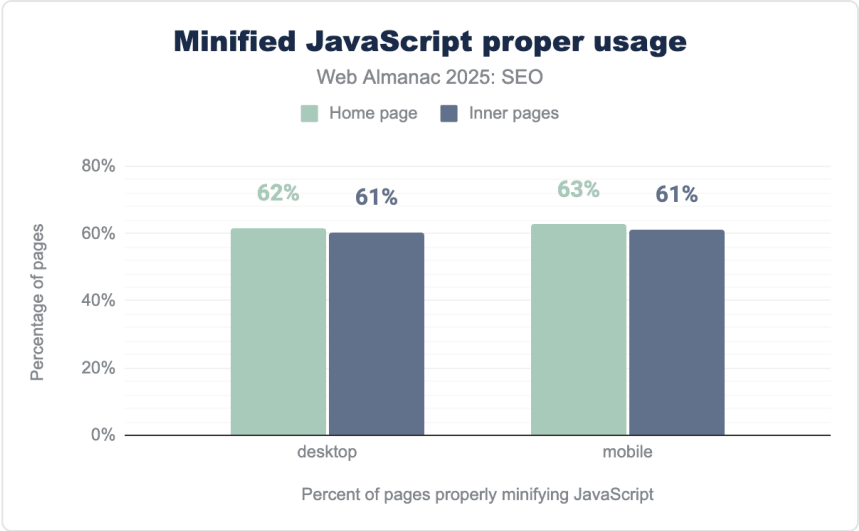


図14.50. JavaScriptの最小化の適切な使用状況。

2025年、デスクトップの62%、モバイルホームページ読み込みの63%がLighthouseのJavaScript最小化[467]テストを通過しました。

内部ページでは、結果は同様で、デスクトップの61%、モバイル読み込みの63%が通過しました。

これは、デスクトップの58%とモバイルホームページの57%が通過し、デスクトップの59%とモバイルの58%の内部ページが同様だった2024年からの改善を示しています。

今年の正しく最小化されたCSSの若干の低下はあるものの、さらに採用の余地がある中で、最小化されているJavaScriptがより多く増加していることは心強いです。

キャッシュ

キャッシュルールの賢明な使用は、過度なページ重量を軽減するのに役立ちます。一般的なリソースがブラウザによってローカルにキャッシュされている場合、訪問者がウェブサイトをナビゲートする際や後で戻ってきた時に再びダウンロードする必要がなくなります。これにより、ネットワークリクエストが減り、ページの読み込みが速くなります。

当然ながら、これは重いリソースの問題を完全に軽減するわけではありません。それらは少なくとも一度はリクエストされてダウンロードされる必要があります。

467. https://developer.chrome.com/docs/lighthouse/performance/unminified-javascript/

キャッシュは変更される可能性が低い静的リソースに最も効果的です。

Lighthouseは効率的なキャッシュ有効期限の使用[468]監査を提供しており、明示的にキャッシュしないよう設定されていない限り、少なくとも30日のキャッシュ有効期限を持つページによって呼び出される静的リソースを探します。

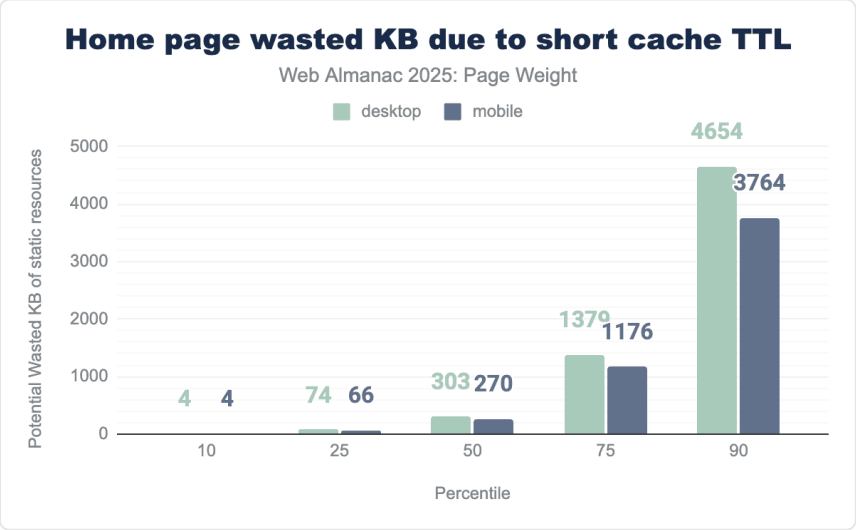


図14.51. 短いキャッシュTTLによるホームページの無駄なKBの分布

468. https://developer.chrome.com/docs/performance/insights/cache

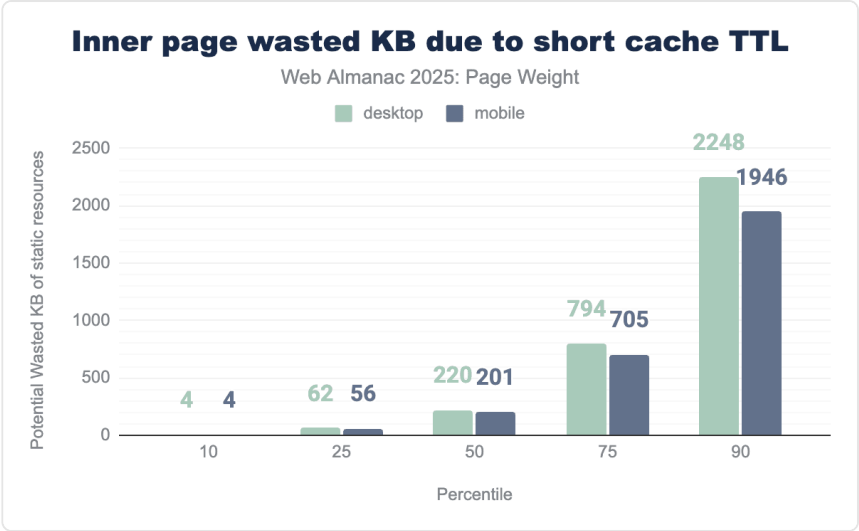


図14.52. 短いキャッシュTTLによる内部ページの無駄なKBの分布

データは、多くのウェブサイトがブラウザキャッシュをより有効活用することでページ重量を削減できることを示しています。デスクトップホームページでは、中央値のサイトが約303 KBを節約でき、モバイルホームページではリソースが一貫してキャッシュから読み込まれた場合に約270 KBを節約できる可能性があります。

内部ページはわずかに優れていますが、まだ改善の余地があります。潜在的な節約の中央値はデスクトップで220 KB、モバイルで201 KBです。

これらの数字は潜在的な節約を表しています。初回訪問者には適用されず、任意のリソースがキャッシュされるかブラウザキャッシュから取得されるという保証もありません。

しかし、潜在的な節約を考えると、キャッシュ戦略を見直す価値があるかもしれません。例えば、キャッシュタイミングを確認し、web.devのHTTPキャッシュを使った不必要なネットワークリクエストの防止[469]記事のガイダンスを見直すことを検討してください。いくつかの調整は、リピートユーザーにとってより速い体験につながり、不必要なネットワーク活動を削減する可能性があります。

CDN

キャッシュと同様に、CDNはデフォルトでファイルサイズを削減しないため、ページ重量の確実な解決策ではありません。代わりに、軽減策として機能します。

469. https://web.dev/articles/http-cache

CDNはオリジンサーバーよりもユーザーに地理的に近いサーバーからコンテンツをキャッシュして提供できます。これはリソースが軽量にはならないものの、物理的な距離が縮まることでコンテンツ配信が速くなり、知覚されるパフォーマンスが向上することを意味します。

多くのCDNは、画像や動画のような資産の圧縮、再エンコーディング、リサイズの処理などの追加機能も提供しています。これらの機能は多くの場合自動または半自動で、つまり最小限の開発時間を必要とし、実際のページ重量を削減するシンプルな方法を提供できます。

これらのページ重量節約最適化テクニックの実装に苦労している場合、すでに使用しているCDNサービスが何を提供しているかを評価するか、他のCDNプロバイダーを探して組み込みツールが何を提供しているか確認することが価値あるかもしれません。 CDNはますます普及し強力になっており、CDNチャプターでより詳細な情報を得ることをお勧めします。

ページの重さとCore Web Vitals

今年、Core Web Vitals[470]データを分析に含めることができました。これらの新しい洞察には重要な注意事項があります：すべてのページがURLレベルのCrUXデータを持っているわけではありません。

明確にするために、CrUXまたはChrome User Experienceデータはウェブサイトをブラウズするオプトインしたchromeユーザーから収集されるデータです。このデータは制御された環境でのパフォーマンスではなく、実際のユーザーがウェブページをどのように体験するかを反映しています。

CrUXデータセットに含まれるためには、ページが最小数の訪問者[471]を駆動する必要があります。その結果、データセットはより人気のあるサイトに偏り、クロールコーパス全体を表していません。

この制限にもかかわらず、常に変化するウェブ上で機能とページ重量のバランスを取る努力をする中で、これらの発見が貴重な視点を提供するものとして役立つことを願っています。

45%

図14.53. 2〜3 MBのモバイルページがCore Web Vitalsに合格

Core Web Vitalsは、ウェブサイトとのインタラクション時の人間体験の品質を測定するために設計された人間中心のメトリクスのセットです。3つの主要な領域に焦点を当てています：

470. *https://developers.google.com/search/docs/appearance/core-web-vitals*
471. *https://developer.chrome.com/docs/crux/methodology#popularity-eligibility*

1. 視覚的な読み込みを測定するLargest Contentful Paint
2. 視覚的な安定性を測定するCumulative Layout Shift
3. インタラクティビティを測定するInteraction to Next Paint

CrUXデータ：重量別Core Web Vitals評価

ページがCore Web Vitals評価に合格するためには、第75パーセンタイルで良好なLCPとCLSメトリクスを示すCrUXユーザーデータを持ち、INPが第75パーセンタイルで良好か完全に存在しないかのいずれかである必要があります。INPはインタラクティビティを測定しており、すべてのページが訪問者を呼び込むわけではないため、INPデータセットは最も疎になりがちです。ページ重量は計算の一部ではありませんが、データは合計メガバイトと合格率の間の明確な相関関係を示しています。

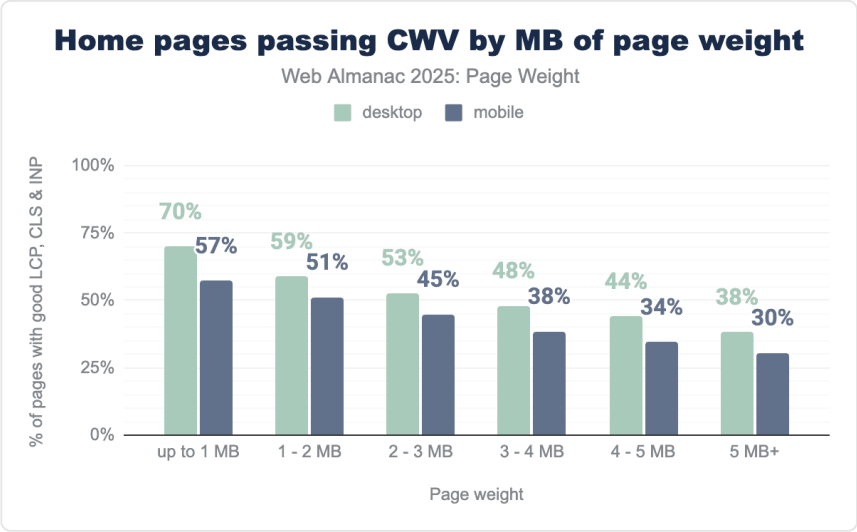


図14.54. ページ重量別のCore Web Vitalsに合格するホームページの割合。

1 MB以下のデスクトップホームページの70%がこの評価に合格しました。対照的に、5 MB以上のホームページは38%しか合格せず、軽量な対応ページの合格率のほぼ半分です。モバイルページは同様のパターンを示しています：1 MB以下のページの57%が合格したのに対し、5 MB以上のページは30%しか合格しませんでした。ホームページの重量が増加するにつれて合格率は着実に低下し、モバイルはすべての重量カテゴリーでデスクトップに一貫して遅れを取っています。

2025年、ホームページの中央値はデスクトップで2.9 MB、モバイルで2.6 MBでした。2〜3 MBの範囲では、デスクトップページの63%がCore Web Vitals評価に合格しました。これらの数字はCrUXフィールドデータに依存しているため、主に最も人気のあるサイトのパフォ

ーマンスを反映しています。同じ重量範囲のモバイルでは、ページの53%が合格しました。

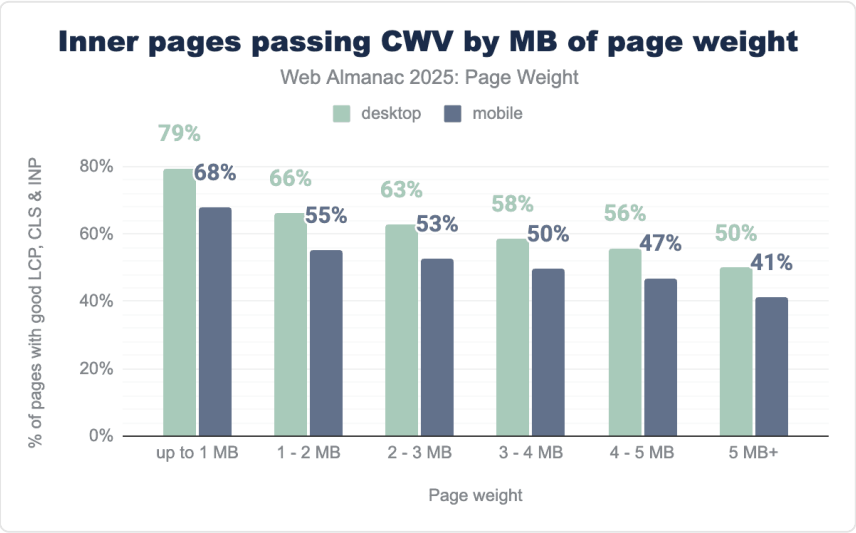


図14.55. ページ重量別のCore Web Vitalsに合格する内部ページの割合。

上記で観察されたように、内部ページはホームページより軽量である傾向があります。2025年、ホームページの中央値はデスクトップで2.9 MB、モバイルで2.6 MBでした。内部ページは比較すると軽量で、モバイルの中央値は1.8 MB、デスクトップは2 MBでした。

データは内部ページの方がホームページよりも緩やかなパターンを示しています。1 MB以下のデスクトップ内部ページの79%がCore Web Vitals評価に合格しました。5 MB以上の内部ページの50%が評価に合格しました。モバイルページは合格しにくく、1メガバイトごとに合格率が徐々に低下します。

CrUXデータ：重量別Largest Contentful Paint

Largest Contentful Paint（LCP）[472]は、ユーザーがページへのナビゲーションを開始してから、ビューポート内の最大の画像、テキストブロック、または動画がレンダリングされるまでにかかる時間を測定します。ユーザー体験の第75パーセンタイルが2.5秒以内に収まると、ページは合格スコアを達成します。

472. *https://web.dev/articles/lcp*

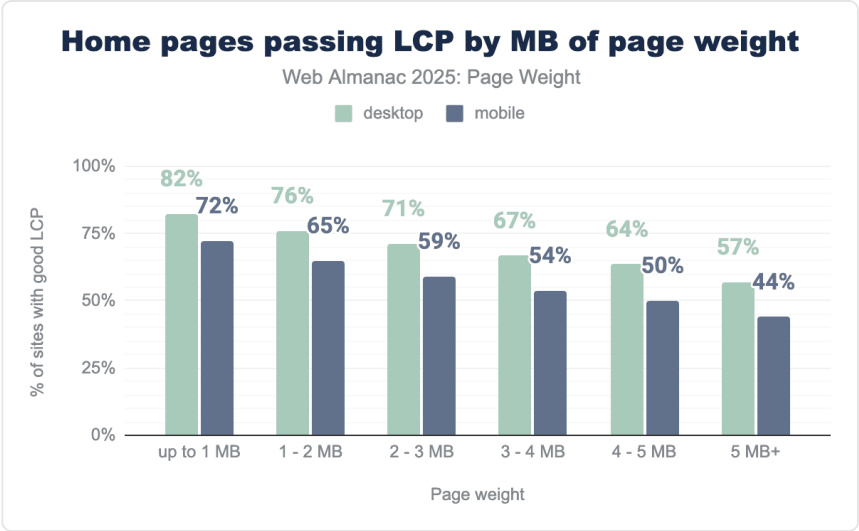


図14.56. ページ重量別の良好なLargest Contentful Paintを持つホームページの割合。

最も軽いページはLCPに合格し、デスクトップの82%、モバイルの72%が合格スコアを達成しました。中央値のページ重量を表す2〜3 MBの範囲のページは、デスクトップで71%、モバイルで59%の合格率でした。最も重いページ、つまり5 MB以上のページでは、合格率はデスクトップで57%、モバイルで44%に低下しました。

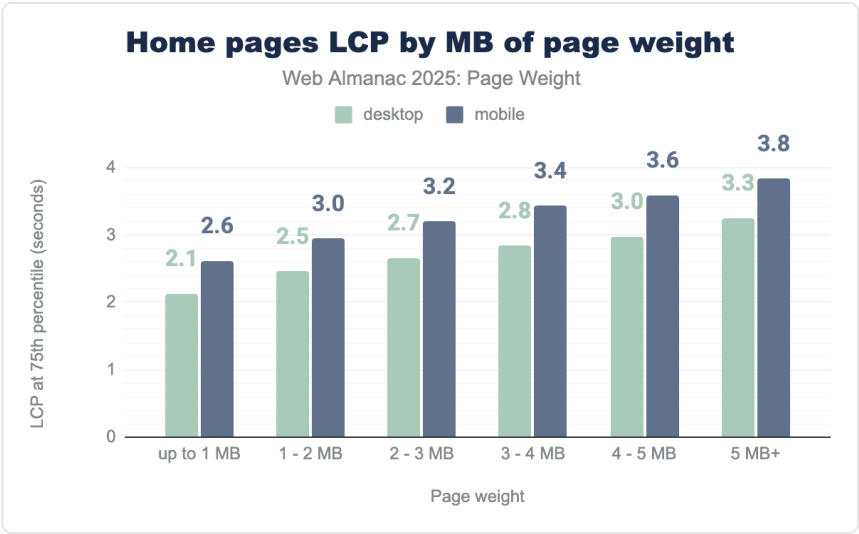


図14.57. ページ重量グループ別の秒単位のホームページのLargest Contentful Paint。

2〜3 MBの範囲のホームページでは、LCPはデスクトップで2.7秒、モバイルで3.2秒で達成されました。モバイルは1 MB以下のページでさえ2.5秒未満を達成するのに苦労し、LCPは2.6秒を記録しました。ページ重量が増加するにつれてLCP時間も相応に上昇し、モバイルは最大コンテンツ要素のレンダリングに約0.5秒長くかかります。

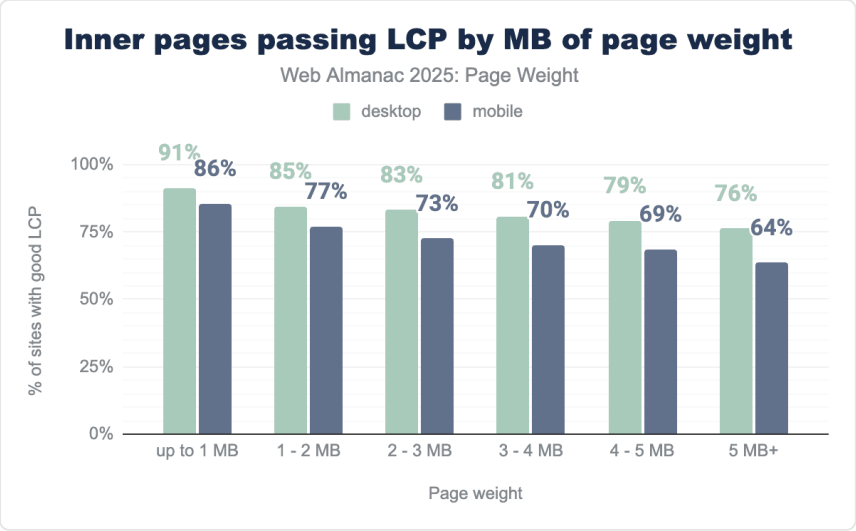


図14.58. ページ重量別の良好なLargest Contentful Paintを持つ内部ページの割合。

内部ページはホームページよりも優れたLargest Contentful Paintのパフォーマンスを示しています。1 MB以下のページでは、デスクトップの91%、モバイルの86%の内部ページがメトリクスに合格しました。

1〜2 MBの範囲のページはモバイル内部ページの中央値サイズを表しており、この範囲のページの77%がスコアを達成しました。2 MBでは、デスクトップの中央値内部ページは同様に動作する2つの範囲の境界に位置しています。2〜3 MBの範囲では、デスクトップ内部ページの83%が合格しました。

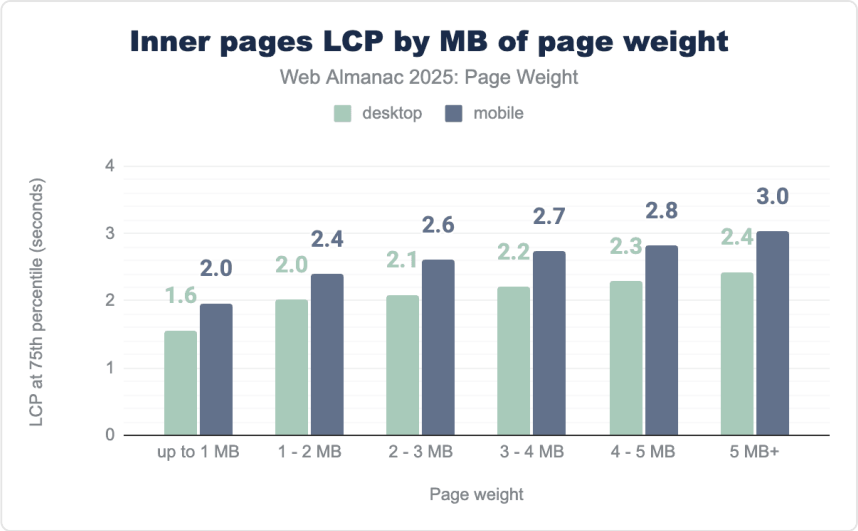


図14.59. ページ重量別の内部ページのLargest Contentful Paint時間。

内部ページは全体的にホームページよりも優れたLCPパフォーマンスを示しています。2 MB以下のページは第75パーセンタイルのLCP合格閾値を満たしました。デスクトップページはすべてのサイズ範囲で一貫して評価に合格しました。

CrUXデータ：重量別Cumulative Layout Shift

2番目のCore Web Vitalsメトリクスであるcumulative Layout Shift（CLS）は、ページの視覚的安定性を測定します。ページ読み込みの最初の5秒間に目に見える要素が予期せず動く量を計算し、移動する要素のサイズと移動距離の両方を考慮します。これらのシフトはセッションウィンドウ全体で平均化されて最終スコアが生成されます。

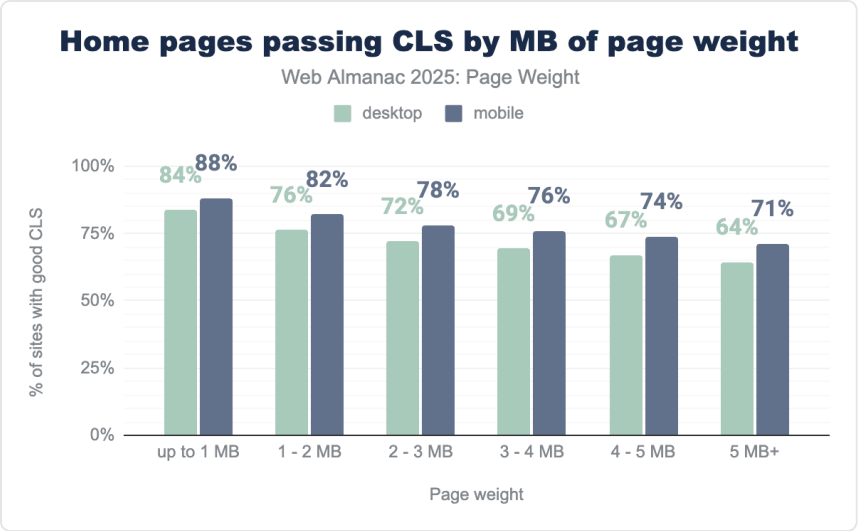


図14.60. ページ重量別の良好なCumulative Layout Shiftを持つ内部ページの割合。

第75パーセンタイルでは、2〜3 MB範囲のデスクトップホームページの72%とモバイルホームページの78%がCLS基準を満たしています。これらのページは2025年の中央値ページを表しています。CLSはLCPよりもパーセンタイル間での低下が少なかった。1 MB以下のモバイルページの88%が合格しました。ページ重量が5 MBを超えてもモバイルページの71%はまだ合格しました。

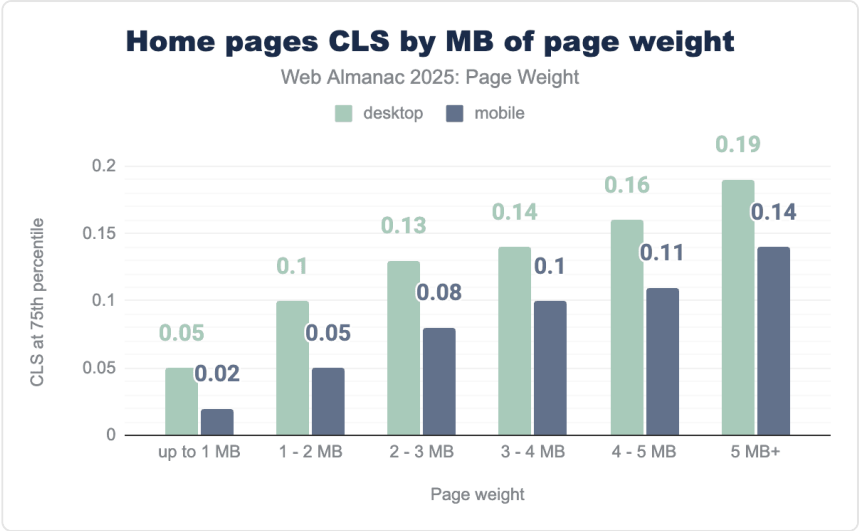


図14.61. ページ重量別のホームページのCumulative Layout Shift。

CLSの合格スコアは≤ 0.10です。CLSスコアはデスクトップで一貫して高く、これはより広いランドスケープが要素のシフトにより多くのスペースを提供するためと考えられます。デスクトップでは2 MB未満のページが評価に合格し、モバイルでは4 MB未満のページが望ましいスコアを達成しました。

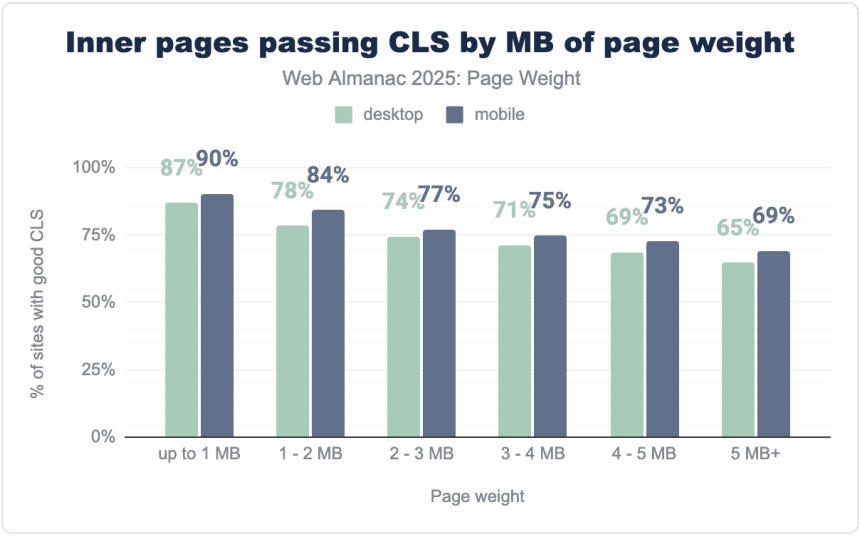


図14.62. ページ重量別の良好なCumulative Layout Shiftを持つ内部ページの割合。

CLSに合格した内部ページの割合はホームページの割合と同様でした。中央値の2〜3 MBを代表するデスクトップモバイルホームページの72%に対し、内部ページの対応する74%が合格しました。モバイル内部ページの77%が合格し、モバイルデスクトップページの78%と比較されました。

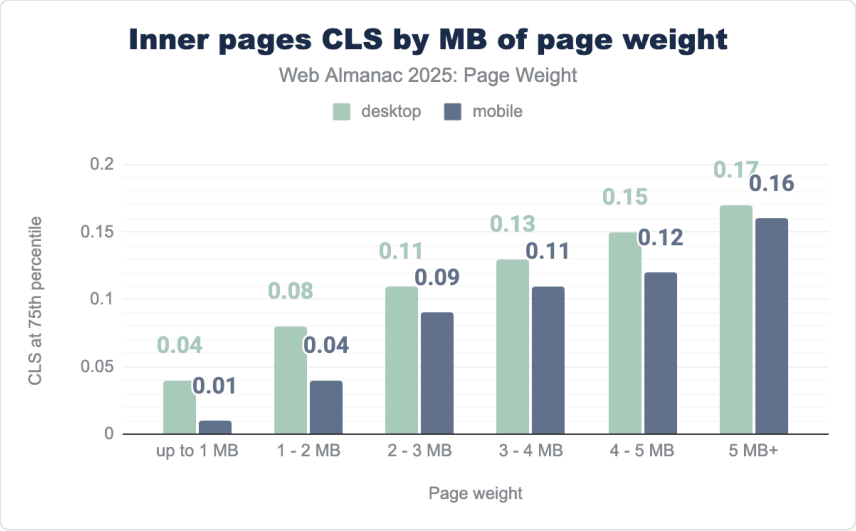


図14.63. ページ重量別の内部ページのCumulative Layout Shift。

内部ページのページ重量がCLSに与える影響は、範囲をまたいで一定のパターンを示しています。2 MB以下のデスクトップページと3 MB以下のモバイルページは0.10未満のスコアを達成できました。3 MB以上のページは一貫して望ましいしきい値を超えるスコアを記録しました。

CrUXデータ：重量別Interaction to Next Paint

3番目のCore Web VitalsメトリクスはInteraction to Next Paint（INP）です。ページのライフサイクル全体でのインタラクティブ性を表すように設計されたINPは、ユーザーがクリック、タップ、またはキーを押してから次の視覚的な更新が表示されるまでのインタラクションの総待ち時間を測定します。200ミリ秒以下のINPは良好なレスポンシブ性を示します。

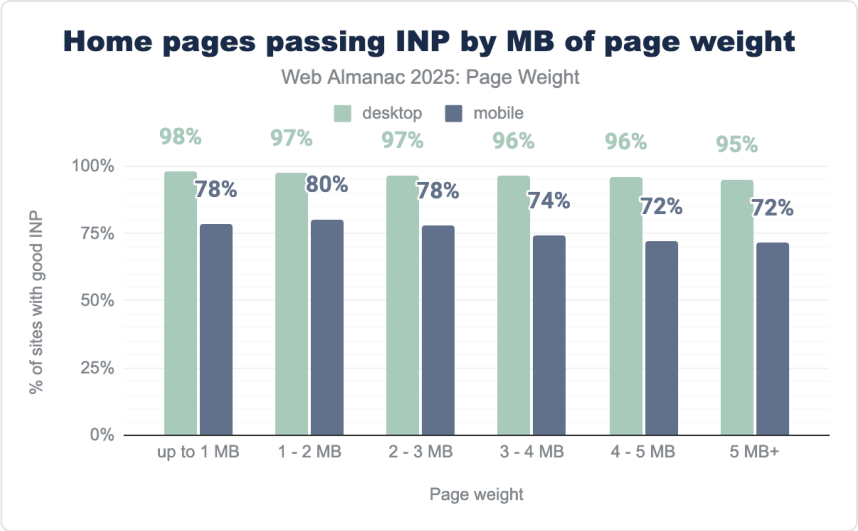


図14.64. ページ重量別の良好なInteraction to Next Paintを持つホームページの割合。

INPメトリクスはすべてのCore Web Vitalsメトリクスの中で最も高いデスクトップ合格率を示しています。1 MB以下のホームページでは98%がINPに合格し、5 MB以上の範囲でも95%のデスクトップホームページが評価に合格しています。

モバイルホームページの合格率は低く、1 MB以下のページの78%が合格しましたが、1〜2 MBの範囲のページはわずかに改善され、80%が200ms未満でINPを達成しました。

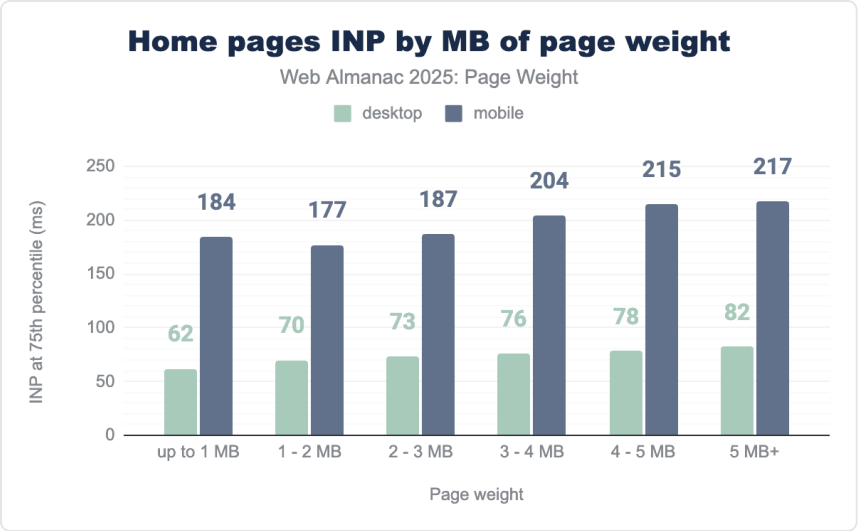


図14.65. ページ重量別のホームページのInteraction to Next Paint。

デスクトップホームページはすべての重量カテゴリで200msの閾値を下回るINP時間を一貫して達成しており、最も重い5 MB以上の範囲でも、第75パーセンタイルのINPはわずか82msでした。3 MB以下のモバイルホームページはINP目標を達成しました。しかし、3〜4 MBの範囲では、第75パーセンタイルのINPが200msを超えました。

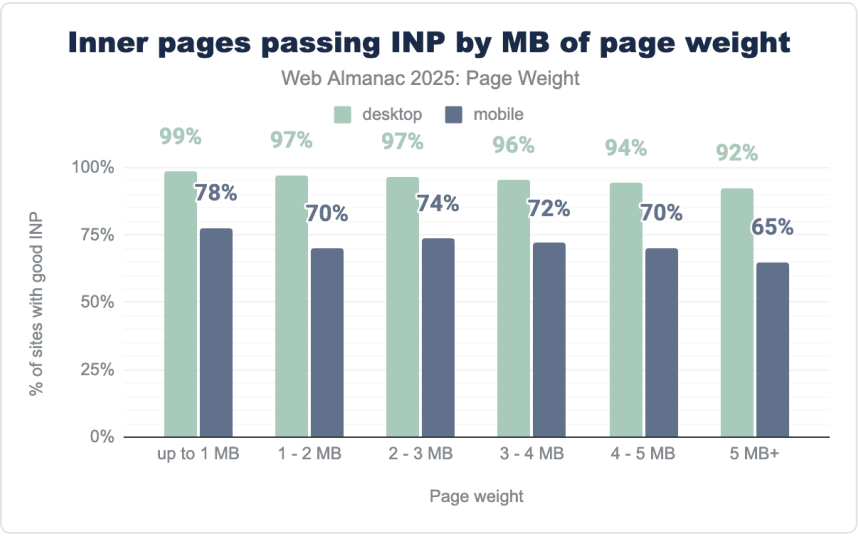


図14.66. ページ重量別の良好なInteraction to Next Paintを持つ内部ページの割合。

内部ページはモバイルとデスクトップの間に同様のギャップを示しています。1 MB以下のページの99%が200ms未満のINPを達成したのに対し、同じ重量範囲のモバイルページは78%にとどまりました。5 MB以上では、デスクトップ内部ページの92%がインタラクティブ性評価に合格したのに対し、65%のモバイルページが目標を達成しました。

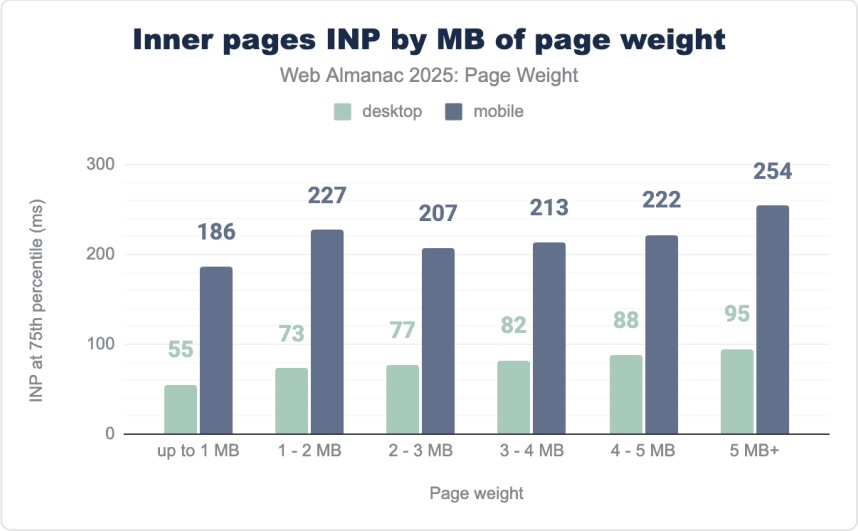


図14.67. ページ重量別の内部ページのInteraction to Next Paint。

Lighthouseデータ：重量別Core Web Vitals評価

上記で分析した新たに利用可能なCrUXデータに加えて、2024年に初めて導入した合成Core Web Vitals評価を継続しています。このデータは、この年次レポートを支えるHTTP Archiveクロール中にLighthouseを実行することで収集されます。Lighthouse[473]は、Googleが開発した無料のオープンソース自動化ツールで、パフォーマンス、アクセシビリティ、SEOなど複数の分野でWebページの品質を評価します。開発者がリリースプロセスの一部として本番環境の問題を監視・軽減するためによく使用されます。

Lighthouseデータをラボデータまたは Syntheticデータとして区別することが重要です。実際の体験をエミュレートするように設計されていますが、実際のユーザーのサイトパフォーマンスに影響するすべての要因を考慮することはできません。また、ページの初期読み込みのみを測定するため、インタラクティブ性の計算が継続的なインタラクティブ性と視覚的安定性に限定されます。

CrUXはリアルな人間の体験をより正確に表していますが、限界もあります。Webページは

473. *https://developer.chrome.com/docs/lighthouse*

データセットに表示されるためにデータ匿名化要件を満たすのに十分な訪問数を受ける必要があり、人気のあるページやウェブサイトが表示される可能性が高くなります。

URLにCrUXデータが利用可能な場合は、よりパフォーマントな体験を作るための取り組みの焦点として考慮すべきです。フィールドデータにより、実際の人間の問題を解決できます。Lighthouseのようなラボデータは、実際のユーザーメトリクスで特定された問題をトラブルシューティングするためのより多くのメトリクスを提供します。

Lighthouseデータ：重量別Largest Contentful Paint

Largest Contentful Paintは、3つのCore Web Vitalsの中でSyntheticテストで最も正確に捉えることができます。出力は、実行する特定のテストシナリオ（コールドロード、事前定義されたスクリーンサイズ、ネットワークスロットリング、CPUスロットリング）に基づく推定値にすぎません。これらの条件がサイトの特定のユーザーにとって現実的かどうかが、LighthouseのLCPがどれだけ「正確」かを決定します。

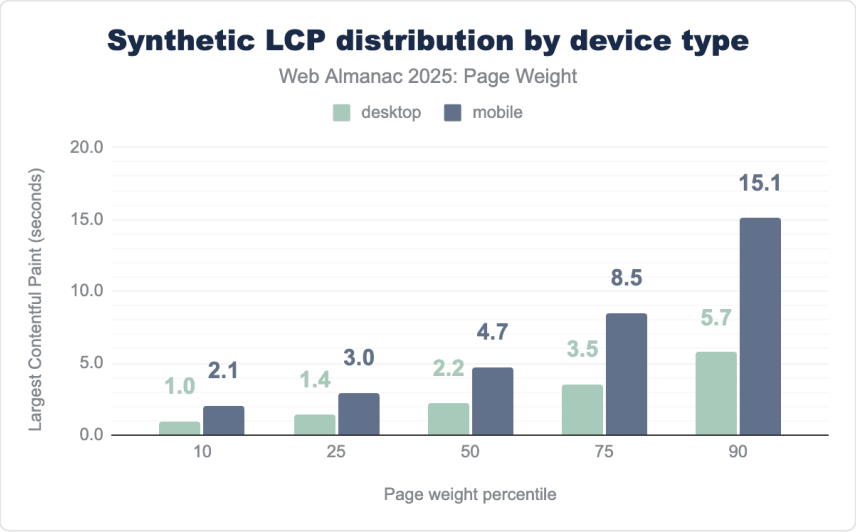


図14.68. デバイスタイプ別のSynthetic Largest Contentful Paint（秒）。

良好なLargest Contentful Paintスコアは2.5秒未満です。Lighthouseテストでは、中央値のデスクトップページがこの閾値を満たし、2.2秒を達成しました。モバイルのパフォーマンスは著しく劣り、同じタスクでLCPに4.7秒かかりました。モバイルページの最速第10パーセンタイルのみが2.1秒のLCPで合格スコアを達成しました。対照的に、第90パーセンタイルのモバイルページは15.1秒のLCPを達成し、推奨目標の約6倍遅い結果でした。デスクトップページは第50パーセンタイルまで2.5秒未満でLCPを達成しました。

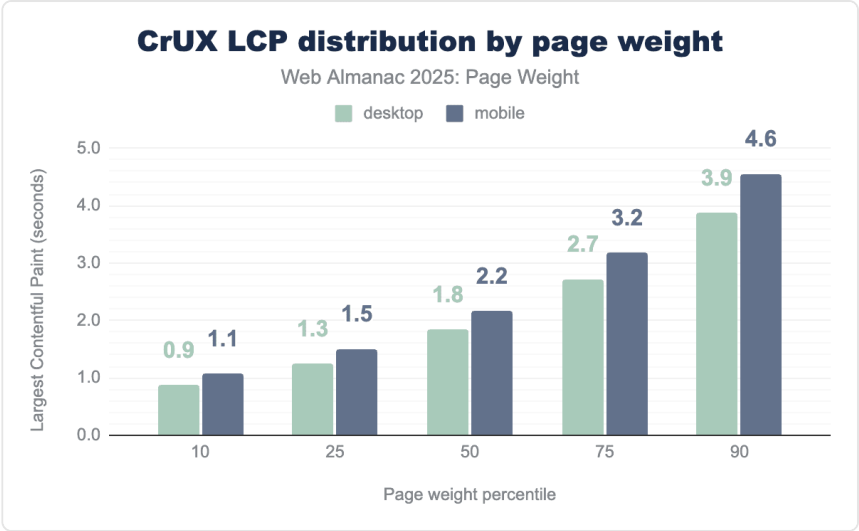


図14.69. CrUXで測定したLargest Contentful Paintのリアルユーザーメトリクス（秒）。

Chrome User Experienceレポートが提供するリアルユーザーメトリクスは、Webのはるかにパフォーマントな姿を描いています。モバイルとデスクトップの両ページが第50パーセンタイルまで2.5秒未満でLCPを達成しました。第90パーセンタイルでは、デスクトップのフィールドデータはラボデータより1.8秒速かった。モバイルはさらに大きなギャップを示し、第25パーセンタイルから始まって、LighthouseはCrUXで見られるものの2倍以上のLCP値を報告し、第90パーセンタイルまでには、Syntheticの結果はリアルユーザーメトリクスより328%高くなりました。

データセットのこのような大きな差異は、このメトリクスの存在理由を裏付けるものかもしれません。LCPが遅いページは、ユーザーが視覚的な読み込みの欠如を無反応と感じるため、放棄される可能性が高くなります。より多くの放棄されたページロードは、CrUXデータセットの対象となるのに十分なページビューを達成する可能性を下げます。

Lighthouseデータ：重量別Cumulative Layout Shift

視覚的安定性はSyntheticテストでは初期ページロード中にのみ測定できます。これはメトリクスの堅牢な性質を正確に反映していません。リアルユーザーのページロードで測定されるCLSは、単一の測定値ではなく、「セッションウィンドウ」内のすべてのレイアウトシフトの集積です。セッションウィンドウは1秒未満の間隔で複数のレイアウトシフトが発生する5秒間の期間です。最も高いスコアのセッションウィンドウがページのCLSスコアとして報告されます。

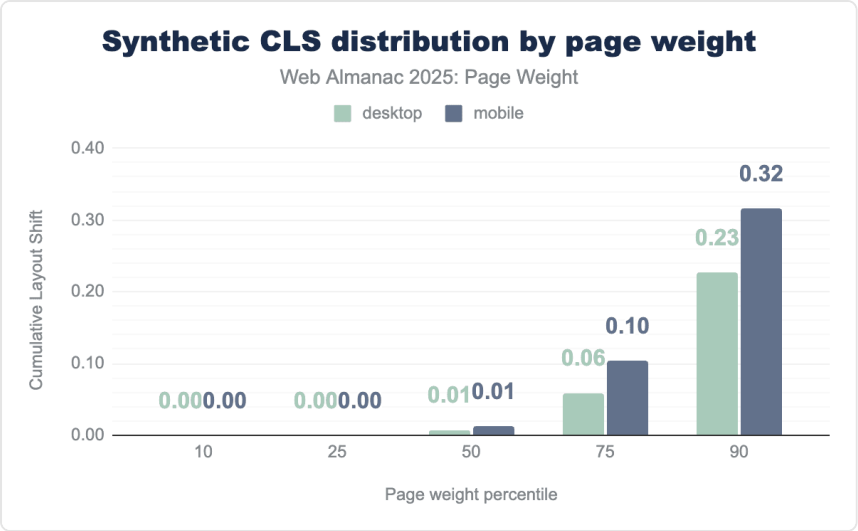


図14.70. デバイスタイプ別のSynthetic Cumulative Layout Shift。

Syntheticデータでは、第75パーセンタイルまでのページが初期ページロードで実質的にCLSに合格していることを示しています。モバイルとデスクトップのスコアは第50パーセンタイルまで一致していました。最も高いスコアは第90パーセンタイルのモバイルページで、0.32を記録しました。

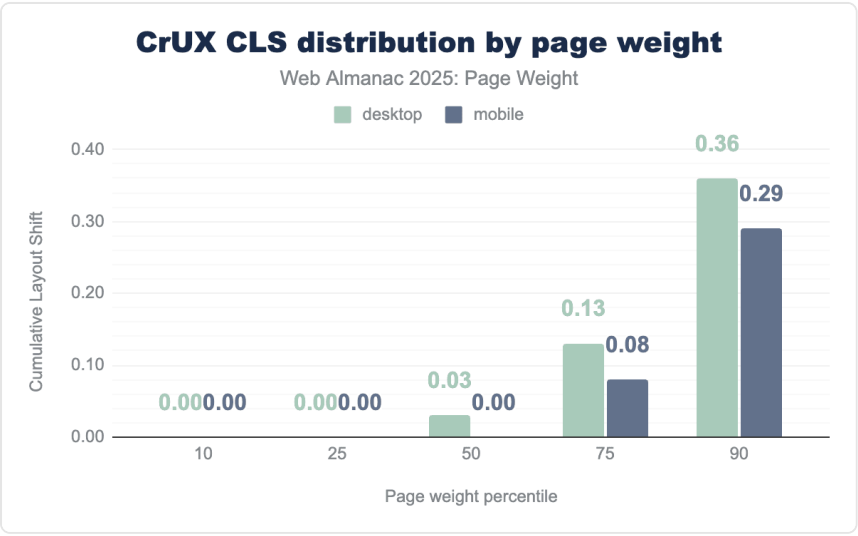


図14.71. CrUXで測定したCumulative Layout Shiftのリアルユーザーメトリクス。

CrUXデータでは、拡張された時間枠とシングルページアプリケーション（SPA）および動的に読み込まれるコンテンツのメカニズムのより良い考慮を導入します。CLSのリアルユーザーメトリクス計算は、ユーザーが体験する可能性が最も高いレイアウトシフトの最大の「バースト」に焦点を当てています。

第50パーセンタイルまでのすべてのユーザーが0.10未満のCLSを体験しました。第75パーセンタイルでは、デスクトップユーザーは良好なCLSを見た一方、モバイルは0.13のCLSでわずかに閾値を超えました。第90パーセンタイルでは、CrUXデスクトップユーザーはモバイルユーザー（0.29）よりも高い視覚的不安定性（0.36）を体験しました。第90パーセンタイルのモバイルユーザーのCrUXデータはSyntheticの対応値（Lighthouseで0.32；CrUXで0.29）より低かった。

Lighthouseデータ：重量別Total Blocking Time

CrUXデータでは、インタラクティブ性はInteraction to Next Paint（INP）として測定されます。つまり、このメトリクスはページのライフサイクル全体でのユーザー入力と視覚的フィードバックの間の時間を測定します。Lighthouseは初期ページロードのみを測定でき、そのためにインタラクティブ性を推定する代替メトリクスを使用します。Total Blocking Time（TBT）[474]は、ページロード中にメインスレッドがブロックされた合計時間を測定するラボメトリクスです。TBTはINPのプロキシとして機能します。高いTBTはしばしば高いINPにつながるため、TBTを修正することがINPを改善する重要な方法です。

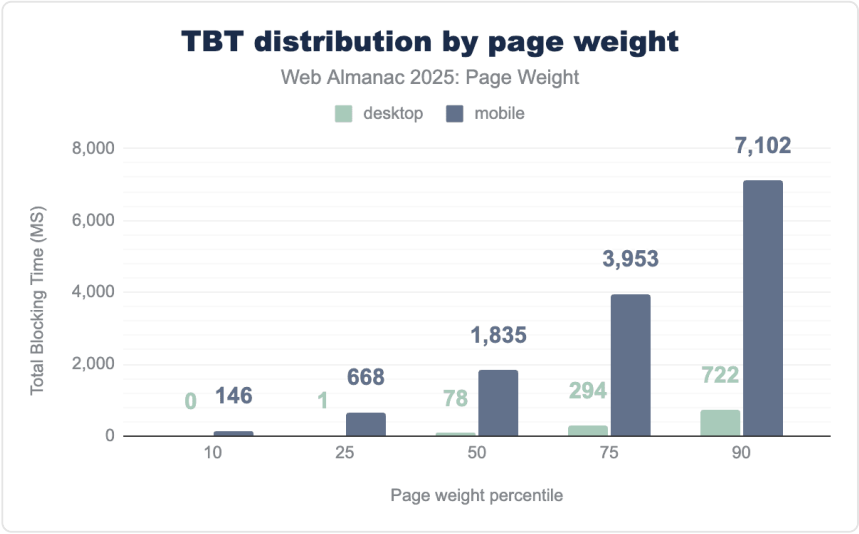


図14.72. Lighthouseで測定したTotal Blocking Time（ミリ秒）。

474. https://web.dev/articles/tbt

メインスレッドが200ミリ秒未満しかブロックされない場合、ページは良好なTBTを持つと見なされます。モバイルページの第10パーセンタイルのみがこのメトリクスを達成しました。デスクトップページは第50パーセンタイルまで200ms未満のTBTを達成しました。第90パーセンタイルでは、モバイルデバイスのメインスレッドは7.1秒間ブロックされました。

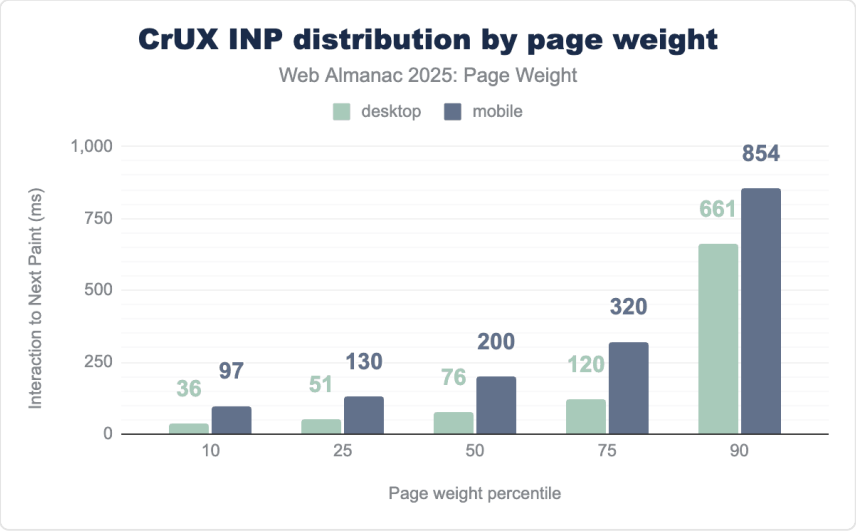


図14.73. CrUXで測定したInteraction to Next Paintのリアルユーザーメトリクス（ミリ秒）。

TBTのフィールドメトリクスの対応値であるINPは、200msという同じタイミング目標を共有しています。CrUXはモバイル体験のよりレスポンシブな姿を描いています。第50パーセンタイルまでのモバイルとデスクトップページは良好なレスポンシブ性を達成しました。第75パーセンタイルでは、デバイス間の差異がより顕著になり、デスクトップのINPは120ms（まだ良好なレスポンシブ性と見なされる）、モバイルでは320msで、ユーザーはより多くのラグを感じます。モバイルCrUXデータの第90パーセンタイルは、200msの目標の4倍のINPを計算しました。Syntheticモバイルデータの第90パーセンタイルは、同じ目標の35倍のTBTを記録しました。

結論

シンプルな結論は、Web上のすべてのページが年々サイズが拡大し続けているということです。この成長はユーザーデバイスとインターネット接続への需要を加速的に増大させています。

機能性対アクセシビリティのバランスは機能性の方向に傾き続けており、接続やデバイスが限られたユーザーはより悪く、包括的でない体験を被っています。

JavaScriptは依然としてこの成長の主要な原動力であり、98.1%のページが少なくとも1つのリクエストを行っています。JavaScriptはしばしば非効率的にページに追加され、ページに重みを加えながらも積極的に使用されていません。

画像とそのホームページと内部ページの両方での広範な使用は、ページに大量の重みをもたらしており、デスクトップとモバイルの差異は昨年に比べて縮小しており、画像がWeb体験に与える影響を考えると最適化への注目が薄れていることが懸念されます。

これらすべての重みの影響は、重みが増加するにつれてCore Web Vitalsの高い失敗率として見ることができ、Chrome User Experienceレポートデータを含め、ユーザーへの影響が理論的ではなく現実のものであることを示しています。

最終的に、ユーザー体験にまったく影響を与えることなく改善の巨大な機会がここにあります。ロスレス圧縮とJavaScriptに必要なものだけを使用することは、ユーザーにとって大きなメリットとWebオーナーのコスト削減を表しますが、現在これらの機会の多くが無視されています。

ページの重さは引き続き重要な問題ですが、現在のところ、解決策からはますます遠ざかっているようです。

著者

Richard Barret

@barkseo.bsky.social rickb110 richardbarrettseo

Richard BarrettはLumar[475]のプロフェッショナルサービスディレクター兼テクニカルサーチコンサルタントです。ウェブオーナーが難しい問題を解決するのを助けることが好きで、いいパズルゲームを2つ3つ好んでいます。

Jamie Indigo

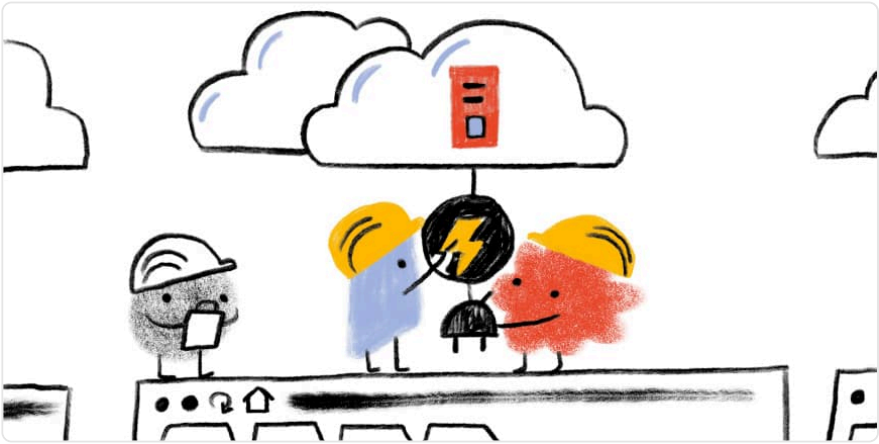
@not-a-robot.com fellowhuman1101 jamie-indigo https://not-a-robot.com

Jamie Indigoはロボットではありませんが、ボットと話します。Cox Automotive[476]のテクニカルSEOディレクターとして、検索エンジンがウェブをクロール、レンダリング、インデックスする方法を研究しています。Jamieはワイルドなアルゴリズムを飼いならしてレンダリング戦略を最適化することが大好きです。仕事以外では、ホラー映画、グラフィックノベル、ダンジョンズ＆ドラゴンズで正義のパラディンを恐怖に陥れることを楽しんでいます。

475. https://www.lumar.io/

476. *https://www.coxautoinc.com/*

部 IV 章 15
CDN



***Joe Viggiano*、 *Alex Moening* と *Nick McCord* によって書かれた。**
Caroline Scholles によってレビュ。
Alex Moening、 *Nick McCord* と *Joe Viggiano* による分析。
Caroline Scholles 編集。
Sakae Kotaro によって翻訳された。

はじめに

本章では、2025年のコンテンツデリバリーネットワーク（CDN）の急速に進化する状況を検証し、HTTPプロトコル最適化における役割に焦点を当てています。CDNは根本的に、HTTP/2多重化によるTCP接続オーバーヘッドの削減からHTTP/3のQUICトランスポートによるヘッドオブラインブロッキングの排除まで、大規模なHTTP配信の課題を解決するために存在しています。HTTPプロトコルがHTTP/1.1の接続制限からHTTP/3の高度な機能へと進化するにつれ、CDNはオリジンサーバーが採用する数年前にこれらのプロトコルを実装する主要な展開手段として機能してきました。

現代のCDNはウェブコンテンツの全スペクトルにわたって配信を最適化しています。高度にキャッシュ可能なリソース（静的アセット、パブリックAPIレスポンス、共有コンテンツ）に対して、CDNは高度な圧縮と最新フォーマット配信によって強化された従来のキャッシュメリットを提供します。キャッシュ可能性が限られたコンテンツ（ユーザー固有のデータ、頻繁に更新されるAPI、パーソナライズされた体験）に対しても、CDNはコンテンツがキャ

ッシュできない場合でも、接続最適化、インテリジェントルーティング、エッジ処理を通じてレイテンシを削減するパフォーマンス改善を提供します。

2024年の包括的な分析を基に、2025年の分析ではプロトコル採用と最適化戦略のシフトが明らかになりました。本章の重要な焦点は、HTTP/3採用の成熟、Server-Timingの透明性などの現代的な最適化技術の台頭、そしてCDNがパフォーマンス、セキュリティ、ユーザーエクスペリエンスに対して取っている洗練された多層的アプローチです。

CDNとは？

コンテンツデリバリーネットワーク（CDN）は、ウェブコンテンツとアプリケーションに高可用性、強化されたパフォーマンス、改善されたセキュリティを提供するために設計された、地理的に分散したサーバーのネットワークです。CDNの主な目標は、エンドユーザーに近い場所からデータを提供することでレイテンシを最小化し、コンテンツ配信を最適化することです。

CDNはエンドユーザーとオリジンサーバー間の中間インフラとして機能し、ウェブリクエストをインターセプトして配信プロセス全体を最適化します。CDNがウェブパフォーマンスをどのように向上させるかを理解するために、ユーザーがブラウザにホスト名を入力する際の従来のウェブインタラクションと、異なるCDNが各ステップをどのように改善するかを考えてみましょう：

- **DNS解決**
    - **従来**: ブラウザがオリジンサーバーのIPのDNSを照会し、解決時間が遅くなることが多い
    - **CDN処理**: CDNのDNSインフラは、様々なルーティング戦略（エニーキャストまたはユニキャスト）を使用してユーザーを最適なエッジサーバーに誘導する場合があります。一部のCDNはHTTPSやSVCB（Service Binding）レコードなどの最新のDNSレコードをサポートしており、DNSレスポンスに直接プロトコル機能をアドバタイズできますが、採用はプロバイダーによって異なります
- **接続確立**
    - **従来**: ブラウザが遠くのオリジンサーバーへの新しいTCP接続を完全なハンドシェイクオーバーヘッドで確立する

    - **CDN処理**: TCP（HTTP/1.1とHTTP/2の場合）またはQUIC付きUDP（HTTP/3の場合）で近くのエッジサーバーに接続。CDNはリピート訪問者向けにHTTP/3の0-RTT接続再開をサポートする場合がありますが、すべてのCDNがこれらの新しい接続最適化機能を実装しているわけではありません

- **プロトコルネゴシエーション**

    - **従来**: オリジンサーバーのプロトコル機能に制限され、多くの場合古いHTTPバージョン
    - **CDN処理**: 多くのCDNは `Alt-Svc`（代替サービス）HTTPヘッダーを通じて、ブラウザに代替プロトコルを通知する最新プロトコルの利用可能性をアドバタイズできます。CDNは通常、オリジンサーバーの機能に関わらず、ブラウザからの新しいプロトコルを受け入れながらオリジンへの接続を維持するプロトコル変換のメリットを提供します

- **リクエスト処理と最適化**

    - **従来**: 最小限の処理による基本的なリクエスト転送
    - **CDN処理**: CDNによっては、ヘッダー正規化、インテリジェントなルーティング決定、サーバーサイドのパフォーマンスメトリクスを提供するServer-Timingなどのパフォーマンスヘッダーの追加、セキュリティヘッダー、コンテンツタイプとユーザーの地理的位置に基づくリクエスト最適化が含まれる場合があります

- **レスポンス処理**

    - **従来**: オリジンサーバーのHTTPサーバー機能に制限された直接レスポンス
    - **CDN処理**: CDNは高度なキャッシュ戦略、キャッシュ検証、Content-Encodingの最適化（BrotliやGzip圧縮など）、条件付きリクエストサポート（帯域幅を節約する304 Not Modifiedレスポンスなど）、レスポンス変換を実装する場合がありますが、具体的な機能はプロバイダーによって異なります

- **接続管理**

    - **従来**: リクエストごとの単一接続またはオリジンへの基本的なキープアライブ
    - **CDN処理**: 多くのCDNは二重接続最適化を実装し、クライアントへの永続的な接続を維持しながら、オリジンサーバーへのインテリジェントな接続プーリングを使用して、両端のオーバーヘッドを削減します

CDNは新興ウェブ標準の展開プラットフォームとして機能し、オリジンサーバーに広く採用される前に、新しいHTTPヘッダー、圧縮アルゴリズム、セキュリティ機能を大規模に実装します。これによりCDNはウェブ技術進化の重要なインフラとして位置づけられますが、利用可能な具体的な機能と最適化はCDNプロバイダーとその技術採用タイムラインに大きく依存します。

注意事項と免責事項

2025年の分析は前年に確立された方法論を基に、新しいメトリクスとより深いパフォーマンス分析を取り入れています。収集された統計は、ベンダー固有のパフォーマンス比較よりも、適用可能な技術と最適化パターンに焦点を当てています。

測定に関する重要事項: この分析におけるすべてのTLSネゴシエーション時間、DNS解決時間、パフォーマンスメトリクスは、制御されたインフラ上でChromeを使用したHTTP Archiveのシミュレートされたブラウザ接続から測定されています。これらの測定値は以下を表しています：

- 一貫したネットワーク条件（制御されたデータセンター接続）
- ChromeブラウザのTLS実装
- セッション再開なしの初回接続
- すべてのCDNにわたる標準化された測定方法論

実際のパフォーマンスは、CDNエッジサーバーへの地理的近接性、ネットワーク条件、TLSセッション再開機能、クライアントデバイスの特性によって異なる場合があります。

テスト方法論の主な制限：

- **シミュレートされたネットワーク条件:** テストは制御されたネットワーク環境を使用

- **単一の地理的視点:** 限られたデータセンターの場所からの分析
- **キャッシュの有効性:** 各CDNは独自の技術を使用しており、多くはセキュリティ上の理由からキャッシュパフォーマンスやキャッシュの深さを公開していません
- **ローカライゼーションと国際化:** 地理的分布と同様に、言語と地理的な特定ドメインの影響もこれらのテストでは不透明です
- **CDN検出:** これは主にDNS解決とHTTPヘッダーを通じて行われます。ほとんどのCDNはDNS CNAMEを使用してユーザーを最適なデータセンターにマッピングしています。ただし、一部のCDNはエニーキャストIPまたは委任されたドメインからの直接A+AAAAレスポンスを使用してDNSチェーンを隠しています。それ以外の場合、ウェブサイトはベンダー間でバランスを取るために複数のCDNを使用しており、クローラーの単一リクエストパスからは隠されています
- **サンプリングバイアス:** HTTP Archiveのデータは人気のあるウェブサイトを反映しており、特定のCDNプロバイダーが過大に代表されている可能性があります
- **市場シェアの解釈:** データはクロールされたサイトにおけるCDN使用パターンを示しており、真の市場分布ではありません

CDNの導入

ウェブページは以下の主要コンポーネントで構成されています：

1. ベースHTMLページ（例：`www.example.com/index.html`—多くの場合`www.example.com`のようなより親しみやすい名前でアクセス可能）。
2. メインドメイン（`www.example.com`）とサブドメイン（例：`images.example.com`、`assets.example.com`）の画像、CSS、フォント、JavaScriptファイルなどの埋め込みファーストパーティコンテンツ。
3. サードパーティドメインから提供されるサードパーティコンテンツ（例：Google Analytics、広告）。

CDN採用パターンの進化は、ウェブアーキテクチャの変化する性質と現代ウェブアプリケーションの複雑さの増大を反映しています。CDNはさまざまなコンテンツタイプにわたってその価値を証明し続けており、異なるユースケースへの適合性を反映したさまざまな採用率を示しています。

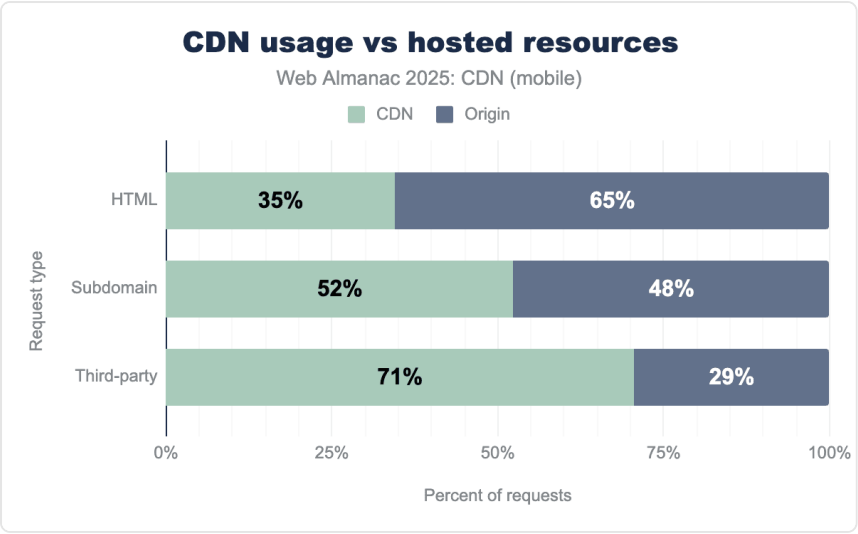


図15.1. モバイルにおけるCDN使用率とホストリソースの比較。

このグラフは、異なるコンテンツタイプ（HTML、サブドメイン、サードパーティ）のリクエストの内訳を示し、モバイルデバイスでCDNとオリジンが提供するコンテンツのシェアを表しています。

CDNはフォント、画像ファイル、スタイルシート、JavaScriptなどの静的コンテンツの配信によく使用されます。この種のコンテンツは頻繁に変更されないため、CDNプロキシサーバーでのキャッシュに適しています。特にサードパーティコンテンツでは71%がCDN経由で配信されるなど、このタイプのリソースにCDNがより頻繁に使用されているのが依然として見られます。サブドメインリソースは52%で中程度のCDN採用を示していますが、HTMLコンテンツはオリジンサーバーから直接配信されることが多いため、35%で最もCDN使用率が低くなっています。

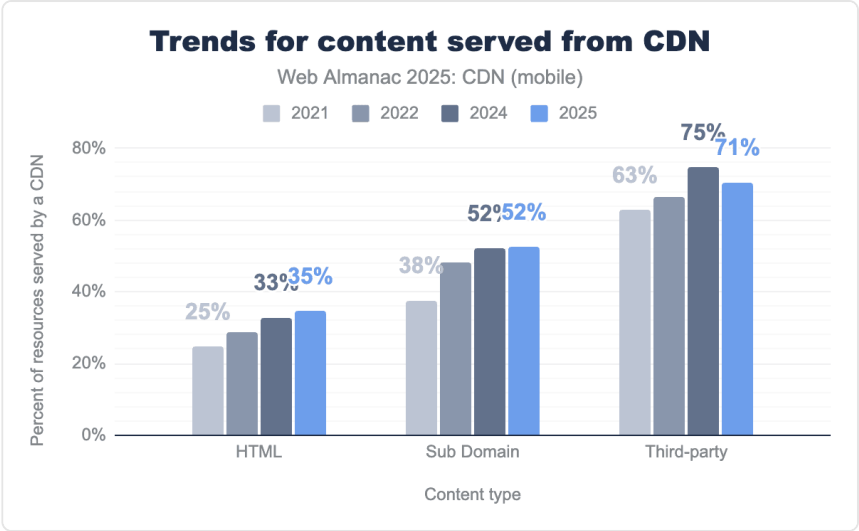


図15.2. モバイルにおけるCDNから配信されるコンテンツのトレンド。

2024年から2025年にかけて、コンテンツタイプにわたってまちまちのトレンドが見られます。HTMLコンテンツは上昇トレンドを続け、33%から35%に増加しました。サブドメインコンテンツは両年で52%と安定していました。しかし、サードパーティコンテンツは2024年の75%から2025年の71%へと顕著に減少し、長年の継続的な成長の後、4パーセントポイントの低下を示しました。

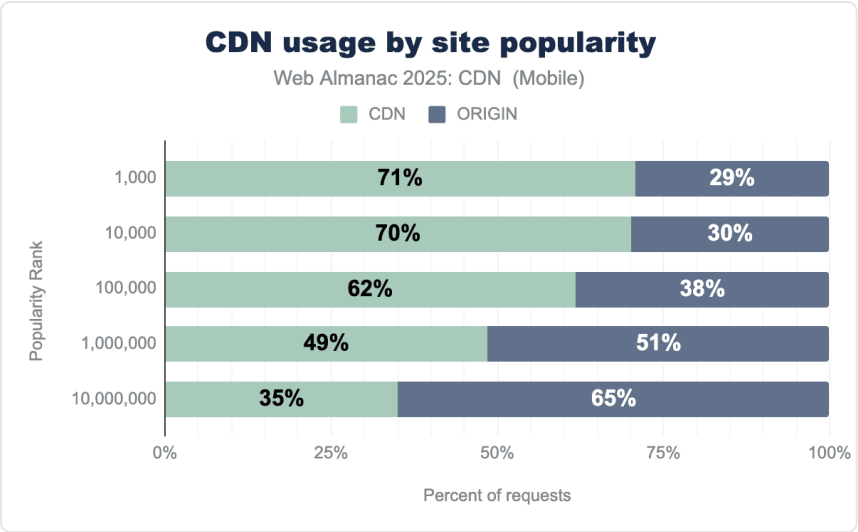


図15.3. モバイルにおけるサイト人気度別CDN使用率。

CDN使用率のシェアは年々増加しており、特にGoogle ChromeのUXレポート（CrUX）分類による最も人気のあるウェブサイトで顕著です。グラフに示されるように、上位1,000ウェブサイトは71%で最も高いCDN使用率を持ち、次いで上位10,000が70%、上位100,000が62%となっています。2024年と比較して、人気ランクに関わらずCDN採用率が増加しています。

過去のエディションで述べたように、小規模サイトにおけるCDN使用率が2024年の33%から2025年の35%に増加したことは、無料またはフラット料金プランや手頃なCDNオプションの台頭に起因していると考えられます。さらに、多くのホスティングソリューションがサービスにCDNをバンドルするようになっており、ウェブサイトがこの技術を活用しやすく、コスト効率の高いものにしています。

CDNプロバイダー

CDNプロバイダーは一般的に2つのセグメントに分類できます：

1. **汎用CDN**: Akamai、Cloudflare、Amazon CloudFront、Fastlyなど、さまざまなユースケースに対応した幅広いコンテンツ配信サービスを提供するプロバイダー。
2. **専用CDN**: NetlifyやWordPressなど、特定のプラットフォームやユースケースに特化したプロバイダー。

汎用CDNは以下を含む広範な市場ニーズに対応しています：

- ウェブサイト配信
- ウェブアプリケーションAPI配信
- 動画ストリーミング
- エッジコンピューティングサービス
- ウェブセキュリティサービス

これらの機能は幅広い業界から支持されており、データにもその傾向が現れています。

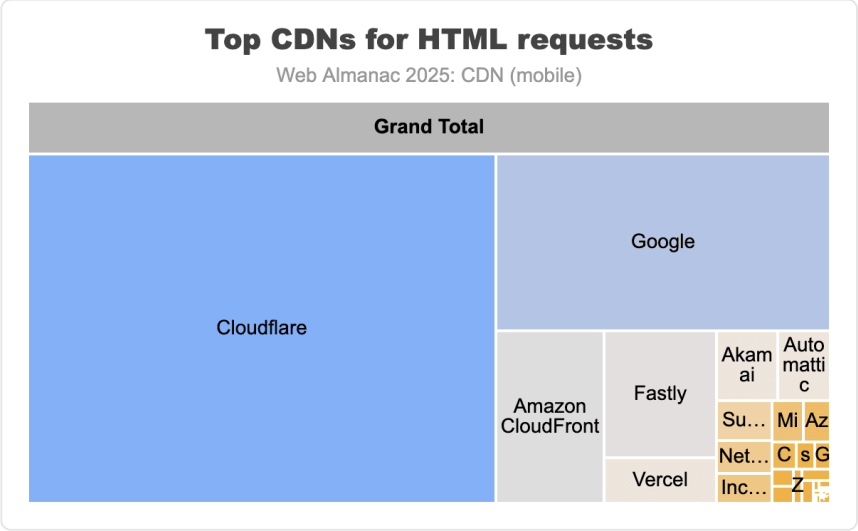


図15.4. モバイルにおけるHTMLリクエストのトップCDN。

ベースHTMLリクエストの処理において、Cloudflareが58%のシェアでトップを占め、次いでGoogle（21%）、Amazon CloudFront（7%）、Fastly（5%）、AkamaiとVercelがそれぞれ2%となっています。

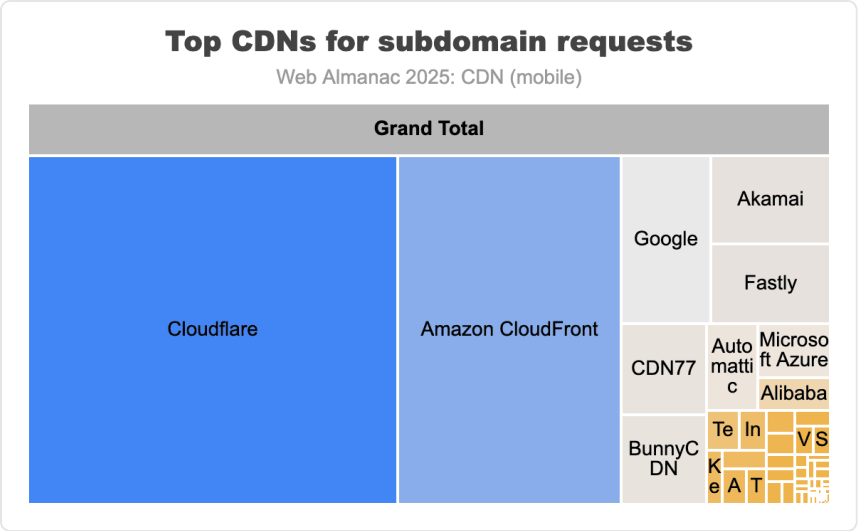


図15.5. モバイルにおけるサブドメインリクエストのトップCDN。

2024年からわずかな変化はあるものの、このカテゴリのトップベンダーはCloudflare（46%）、Amazon CloudFront（28%）、Google（5%）、Akamai（4%）となっています。

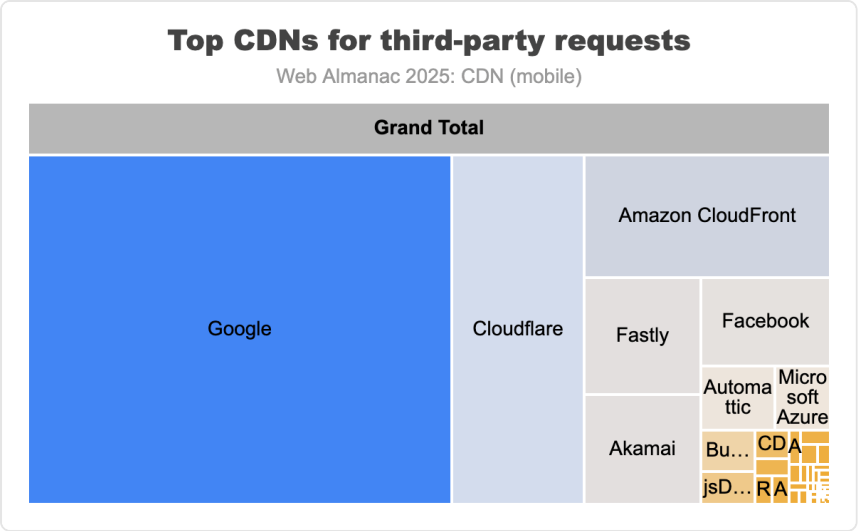


図15.6. モバイルにおけるサードパーティリクエストのトップCDN。

Googleがサードパーティドメイン使用において53%の市場シェアでリストをリードし、次いでCloudflare（17%）、Amazon CloudFront（11%）、Fastly（5%）、AkamaiとFacebook（4%）などの有名なCDNプロバイダーが続いています。

多くのCDNはコンテンツ配信に特化した機能を提供していますが、これらはますます大規模なサービスエコシステムの一部として存在するようになっています。これらのCDNはクラウドインフラ、セキュリティソリューション、エッジコンピューティングプラットフォームと緊密に統合されており、これらの隣接サービスはCDN自体を通じて、またはCDNと並行して提供されています。

異なるCDNプロバイダーは最適化と専門化に対して異なるアプローチを取っています。GoogleやFacebookなどのサードパーティプラットフォームは、自社のニーズに特化して設計された高度に専門化されたCDNを構築し、広告配信のための大規模なスループットを処理し、大規模にアナリティクスビーコンをキャプチャしています。これに対し、CloudflareやAmazon CloudFrontなどの汎用CDNは、より広い適用性を維持しながら特定の機能セットを最適化しています。これらのプラットフォームはCDN機能をマネージドサービスの基盤として活用し、グローバル分散APIゲートウェイやクライアントサイドのデバイスフィンガープリンティングとセキュリティ検査のためのリアルタイムJavaScript注入などのユースケースを可能にしています。

HTTP/3の採用

IETFが2022年6月に発行したHTTP/3[477]はHTTP/2に続くHTTPネットワークプロトコルの大幅な改訂版です。HTTP/3への移行はウェブの歴史上、最も重要なプロトコルアップグレードの1つを表しています。QUICトランスポート上に構築されたHTTP/3は、ヘッドオブラインブロッキングを排除し、接続確立のオーバーヘッドを削減します。2025年の分析では、特にCDNプロバイダーにおける採用パターンの劇的な変化が明らかになりました。

HTTP/3のパフォーマンス改善は、CDNがグローバルスケールで活用するのに独自の立場にある根本的なプロトコル設計の変更から生まれています。HTTP/2がTCPに依存しているのとは異なり、HTTP/3はUDP上のQUICを使用し、TCPのヘッドオブラインブロッキングを排除しています。1つのHTTP/2ストリームがパケットロスに遭遇すると、そのTCP接続上のすべての多重化ストリームが停止します。HTTP/3を実装したCDNは、QUICの独立したストリーム回復を通じて影響を受けないリクエストのパフォーマンスを維持できます。これは、低速の画像読み込みが重要なCSSやJavaScriptの配信をブロックしない、混合コンテンツを提供するCDNにとって特に価値があります。

HTTP/3は、QUICの統合された暗号ハンドシェイクを通じて、HTTP/2の3 RTT（TCPハンドシェイク、TLSハンドシェイク、HTTPネゴシエーション）から1 RTTへ接続確立を削減しま

477. *https://datatracker.ietf.org/doc/html/rfc9114*

す。CDNは地理的近接性を通じてこの利点を増幅させ、近隣のCDNエッジサーバーに接続するユーザーは接続時間の短縮を体験します。リピーターのユーザーに対して、一部のCDNは0-RTT接続再開を実装しており、同じエッジサーバーへの再接続時にハンドシェイクのオーバーヘッドを完全に排除します。

現代のCDNは、DNSレスポンスがHTTP/3の可用性と接続パラメータを直接広告できるService Binding（SVCB）/ HTTPSレコードの一種であるHTTPS DNSレコードを実装し始めています。ブラウザがCDN対応ドメインのDNSをクエリすると、HTTPSレコードは特定のQUICパラメータとともにポート443でのHTTP/3サポートを示すことができ、従来のHTTP/2からHTTP/3へのアップグレードプロセスなしに即時HTTP/3接続を可能にします。このDNSレベルの最適化は、プロトコルアップグレードのネゴシエーションを排除し接続確立時間を短縮できるため、複数のドメインとサービスを管理するCDNにとって特に強力です。ただし、HTTPSレコードの実装はCDNプロバイダーによって大きく異なり、先行実装しているものとまだサポートを実装していないものがあります。

方法論に関する重要な考慮事項

HTTP/3採用率の測定はHTTP Archiveの方法論の特定の特性を反映しており、インターネットトラフィック全体の代表値として解釈すべきではありません：

- **サイト選定**: 分析は現代のプロトコルを実装する可能性が高い人気ウェブサイト（上位1,000万）に焦点を当てています
- **リソースタイプへの焦点**: 測定は主にメインドキュメントの読み込みではなく、CDNが提供する静的リソース（CSS、JS、画像）を反映しています
- **地理的範囲**: 限られたデータセンターロケーションからのテストはグローバルなユーザー体験を反映しない可能性があります
- **外部バリデーション**: Cloudflare自身の2025年データはグローバルでの HTTP/3採用率を約21%と報告しており、私たちのデータセットでCloudflareリソースに観察された69%よりも大幅に低くなっています

これらの測定は主要なウェブプロパティとCDN提供リソースにおけるプロトコル採用を理解するために価値がありますが、一般的なインターネット使用パターンを代表するものとして外挿すべきではありません。

CDNプロバイダーにおけるHTTP/3の採用

2025年に、過去年度と一貫した方法論を使用し、CDN提供リソースにおける有意なHTTP/3採用が観察されました。これらの数値は私たちのデータセットのサイトとリソースの特定の

サブセットを反映していますが、CDNプロトコル展開における明確なパターンを示しています：

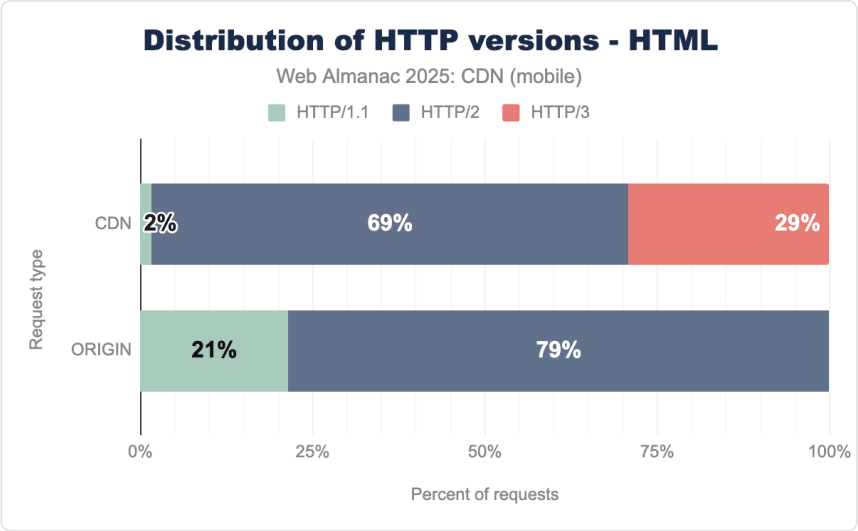


図15.7. HTMLのHTTPバージョン分布（モバイル）。

私たちのデータセット内では、データは人気ウェブサイトと静的リソースにおける新しいプロトコルの採用を推進するCDNの役割を裏付けています。CDNはHTTP/3のトラフィックが29%を占め、オリジントラフィックでは実質的に0%でした。2024年と比較して、2025年はCDNでのHTTP/1.1使用が大幅に減少しました。

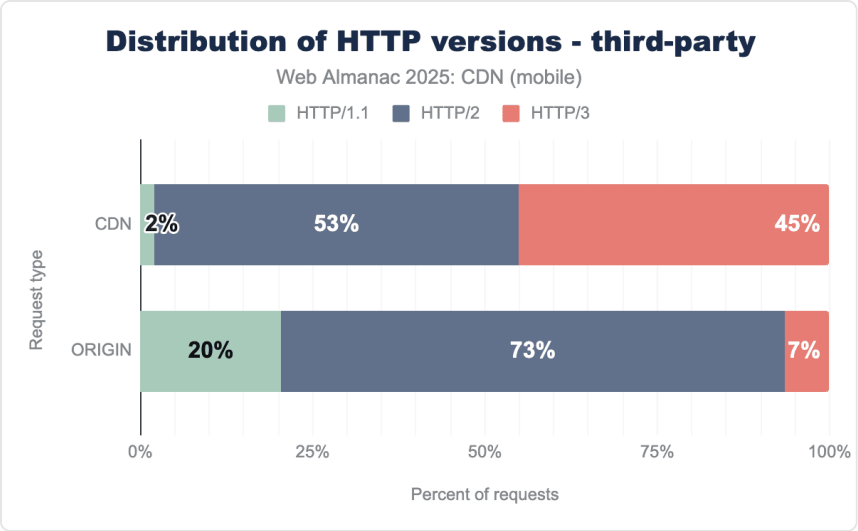


図15.8. サードパーティリクエストのHTTPバージョン分布（モバイル）。

過去数年間、CDNでのHTTP/1.1使用は継続的に減少しているが、2025年にはCDNでのHTTP/1.1使用が2024年のHTMLリクエストの16%からわずか2%へと大幅に減少しました。この減少はオリジンリクエストでさらに顕著で、2024年の56%のHTTP/1.1リクエストから2025年の21%へと低下しました。

ドキュメント（多くの場合、最初のリクエスト）における個々のCDNのHTTP/3サポートを見ると、以下のようになります：

CDN	デスクトップ	モバイル
Cloudflare	44.0%	49.9%
CDN	6.2%	5.0%
Amazon CloudFront	3.2%	3.8%
Sucuri Firewall	0.5%	0.6%
Akamai	0.2%	0.2%
QUIC.cloud	0.1%	0.1%
Google	0.1%	0.1%

図15.9. HTTP/3で提供されたドキュメントリクエストのトップ率。

ここで業界をリードしているCloudflareを特に称えたいと思います。

サブリソースリクエストは、後続リクエストがHTTP/3で行われる可能性が高い複数のリクエストに使用されることが多いため、採用率がはるかに高くなっています：

CDN	デスクトップ	モバイル
Facebook	99.4%	99.9%
Nexcess CDN	97.7%	99.6%
Sucuri Firewall	71.2%	68.4%
Cloudflare	68.6%	69.5%
Reapleaf	59.5%	59.1%
Erstream	31.5%	77.4%
QUIC.cloud	48.8%	47.3%
Google	39.6%	42.2%
Pressable CDN	40.1%	40.8%
Automattic	25.6%	29.0%

図15.10. HTTP/3で提供されたドキュメントリクエストのトップ率。

CDNのパフォーマンス

CDNのパフォーマンスは単にコンテンツをユーザーに近い場所にキャッシュすることを超えています。CDNはブラウザが接続を確立してデータを受信する速さを決定する基礎的なプロトコルと接続メカニズムを積極的に最適化し、現代のウェブアプリケーションのボトルネックを理解するための透明なメトリクスを提供します。パフォーマンス最適化技術には接続プーリング、プロトコル変換、インテリジェントルーティングが含まれ、これらすべてがレイテンシの削減とユーザー体験の向上に貢献します。

HTTP/3 TTFBパフォーマンス

以下は主要CDNにわたるHTTP/3、HTTP/2、HTTP/1.1のレイテンシの中央値パーセンタイル分布です。これらの測定はシミュレートされたブラウザ接続を反映しており、実際のパフォーマンスとは異なる場合があります。

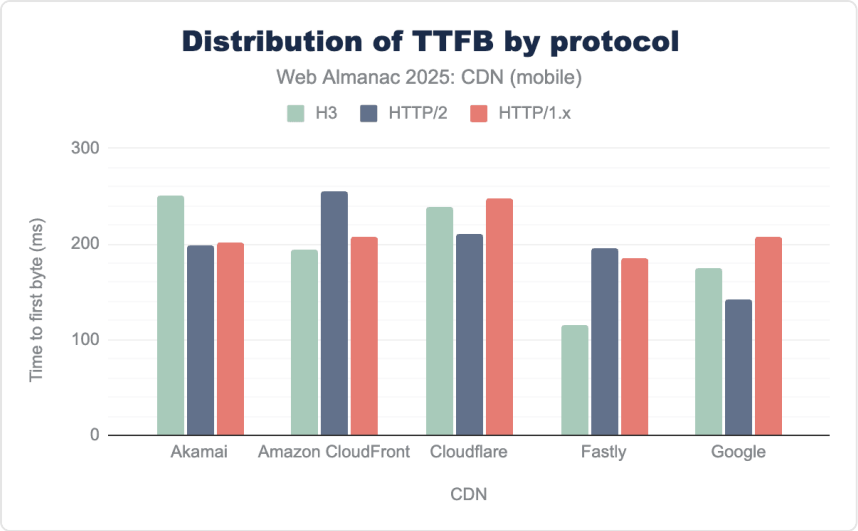


図15.11. 最初のバイトまでの時間（TTFB）の分布（モバイル）。

以前のHTTPプロトコルバージョンと比較したHTTP/3のTTFBパフォーマンスはCDNプロバイダーによって異なり、Amazon CloudFrontとFastlyの両方が改善されたTTFBレイテンシを示しています。これらはCDN間で均一なテストではなく、ウェブサイトオーナーは独自の管理されたパフォーマンステストを実施すべきです。

DNSパフォーマンス

DNS解決はあらゆるウェブリクエストの重要な最初のステップです。2025年のデータは、オリジンサーバーに対するCDNの大幅なパフォーマンス優位性を示しています：

DNS解決パフォーマンス（中央値）：

- **CDN（モバイル）**: 176ms
- **オリジン（モバイル）**: 217ms
- **CDNの優位性**: 19%高速

- **CDN（デスクトップ）**: 52ms
- **オリジン（デスクトップ）**: 129ms
- **CDNの優位性**: 60%高速

デスクトップのパフォーマンス優位性は特に顕著で、CDNはオリジンサーバーより2.5倍高速にDNSレスポンスを提供します。この改善はCDNのエニーキャストルーティング、広範なエッジネットワーク、最適化されたDNSインフラの使用から生まれています。

Alt-Svc

Alt-Svc[478]（代替サービス）HTTPレスポンスヘッダーは、同じリソースにアクセスするために使用できる代替プロトコルやサーバーについてブラウザに通知します。最も一般的なユースケースはHTTP/3サポートのアドバタイズです。ブラウザが最初にHTTP/1.1またはHTTP/2を使用してサーバーに接続すると、サーバーは次のようなAlt-Svcヘッダーを含めることができます：`Alt-Svc: h3=":443"; ma=86400`

これはブラウザに、ポート443でHTTP/3を使用して同じサービスにアクセスできること、およびこの情報が86400秒（24時間）有効であることを通知します。ブラウザがこのヘッダーを受信すると、HTTP/2から始めてアップグレードをネゴシエートするのではなく、HTTP/3を直接使用して将来の接続を確立できます。

2025年のデータでは：

- **Google HTTP/2 → H3アドバタイズ**: 99.2%（HTTP/3をアドバタイズする4,400万のHTTP/2リクエスト）
- **Cloudflare HTTP/2 → H3アドバタイズ**: 100%（HTTP/3をアドバタイズする4,400万のHTTP/2リクエスト）
- **Amazon CloudFront HTTP/2 → H3アドバタイズ**: 100%（HTTP/3をアドバタイズする2,800万のHTTP/2リクエスト）
- **オリジンサーバー HTTP/2 → H3アドバタイズ**: 99.89%（HTTP/3をアドバタイズする5,400万のHTTP/2リクエスト）

478. *https://datatracker.ietf.org/doc/html/rfc7838#section-3*

このデータは、HTTP/2を使用するCDNがAlt-Svcヘッダーで HTTP/3の可用性をほぼ普遍的にアドバタイズし、ブラウザが後続リクエストでHTTP/3にアップグレードできることを示しています。この広範なプロトコルアドバタイジングは、ウェブ全体でHTTP/3のより広い採用を促進するのに役立ちます。

Server-Timing

W3C Server-Timing仕様で定義された `Server-Timing` HTTPヘッダーは、サーバーがリクエスト処理に関するパフォーマンスメトリクスをブラウザに通知できるようにします。このヘッダーはパフォーマンスメトリクスを直接通信し、不透明なサーバー処理を観察可能でデバッグ可能なデータに変換します。CDNに特有のこととして、`Server-Timing` ヘッダーは追加の監視インフラを必要とせずに、CDNエッジ処理時間、オリジンフェッチ時間、またはキャッシュステータスへの透明性を提供するのに役立ちます。

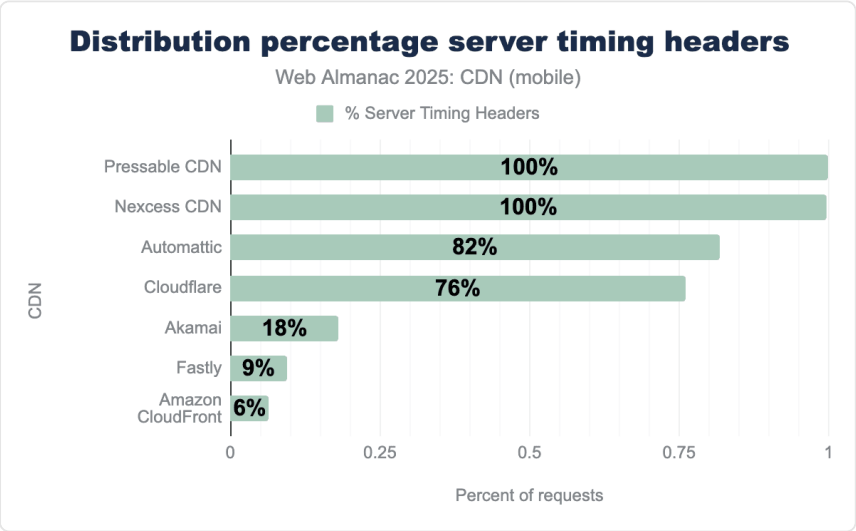


図15.12. サーバータイミングヘッダーの分布割合（モバイル）。

`Server-Timing` ヘッダーの採用はCDN間で異なります。PressableとNexcess CDNはデフォルト設定によりリクエスト全体で100%の採用率を示しています。Akamai、Amazon CloudFront、Fastlyなどのような CDNはデフォルト以外の設定が必要で、採用率が低くなる可能性があります。ただし、セキュリティ、プライバシー、パフォーマンスに関するエンタープライズの懸念がこのオプトインアプローチを推進している可能性があります。

CDNセキュリティヘッダー

CDNは、一般的な攻撃からユーザーを保護するセキュリティヘッダーを実装・適用することで、ウェブセキュリティにおいて重要な役割を果たしています。HTTP Strict Transport Security（HSTS）、X-Frame-Options（XFO）、Content Security Policy（CSP）などのセキュリティヘッダーは、中間者攻撃からクリックジャッキング、クロスサイトスクリプティングまであらゆる脅威を防ぐのに役立ちます。CDNはユーザーとオリジンサーバーの間に位置するため、オリジンが提供するものに関係なくこれらのヘッダーを挿入または変更でき、サイト運営者がセキュリティのベストプラクティスを展開しやすくします。

セキュリティの実装はCDNプロバイダーによって大きく異なり、デフォルト設定と設定可能性に関する異なる哲学を反映しています。一部のプロバイダーはセキュリティヘッダーを自動的に注入しますが、他のプロバイダーは明示的な設定を必要とし、セキュリティと柔軟性のバランスを取っています。

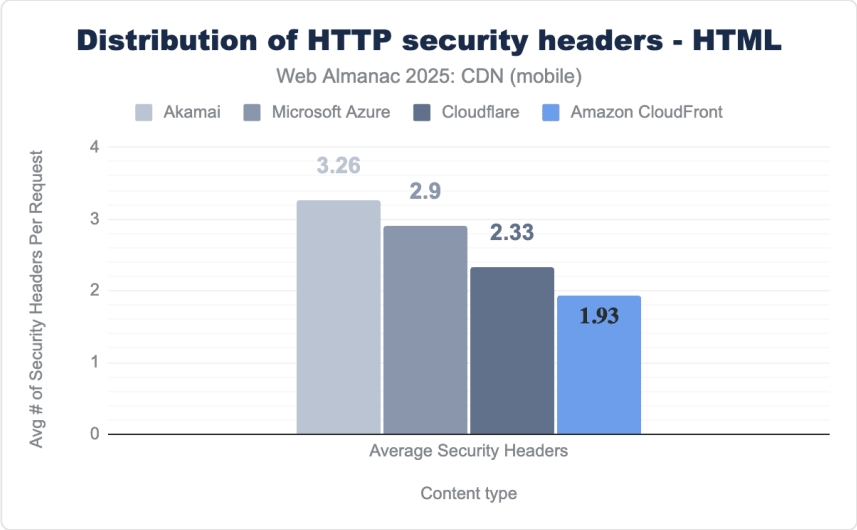


図15.13. HTTPセキュリティヘッダー数の分布（モバイル）。

CloudflareとAmazon CloudFrontはどちらもセキュリティヘッダーの平均数が少なく、以下に示すように特定のヘッダーをより詳細に見てもこのトレンドは続いています。ヘッダー数はセキュリティ態勢の直接的な代理指標ではなく、CDNが自動注入する量と明示的な設定が必要な量を反映していることが多いです。

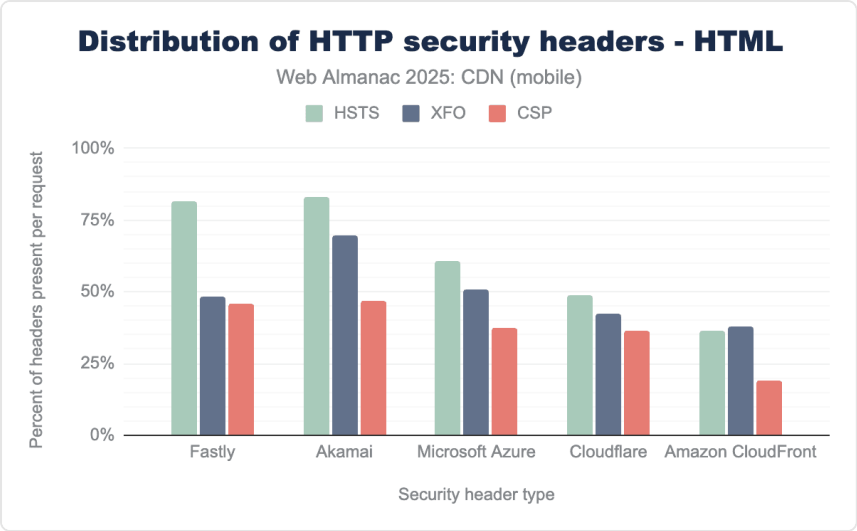


図15.14. HTTPセキュリティヘッダーの分布（モバイル）。

FastlyとAkamaiは基本機能が有効化された際にセキュリティヘッダーのより安全なデフォルトを持っており、セキュリティヘッダーの高い採用率を促進しています。Amazon CloudFrontとCloudflareはセキュリティヘッダーを注入・適用するためにより多くのデフォルト以外の設定が必要で、採用率が低くなっています。

圧縮

圧縮はウェブコンテンツ配信に不可欠であり続け、ウェブサイトで利用可能な最も手軽なパフォーマンス最適化の1つを代表しています。ファイルサイズが小さくなることは、ページの読み込みが速くなり、帯域幅コストが低下し、ユーザーにとってより良い体験となることを意味します。ネットワーク速度が向上し接続オプションが拡大しても、圧縮はすべての種類のインターネット接続のパフォーマンス最適化において重要であり続けます。

現代の圧縮アルゴリズムは従来の方法に比べて大幅な改善を提供しており、CDNがこれらの高度な技術の採用をリードしています。圧縮アルゴリズムの選択は、圧縮率、CPU使用量、互換性要件のトレードオフを伴います。

2025年のデータセットから、一般的に使用されるいくつかの圧縮アルゴリズムが観察されました：

- **Gzip**: レガシースタンダード。広く互換性があるが効率は低い

- **Brotli**: Gzipより20〜26%優れた圧縮を提供する現代的アルゴリズム
- **Zstandard（Zstd）**: 優れた圧縮率と速度を持つ新興スタンダード

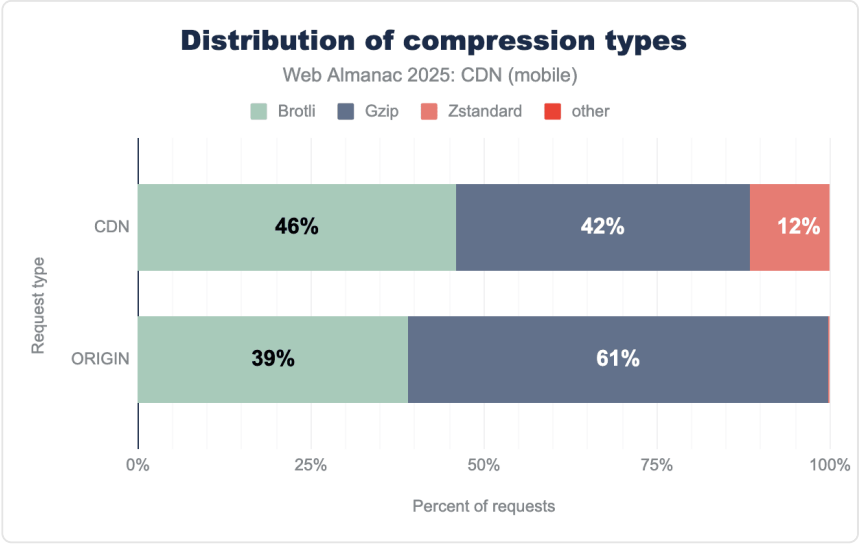


図15.15. 圧縮タイプの分布（モバイル）。

CDNはBrotliとZstandard圧縮の採用をリードしています。2024年と比較して、Zstandardは2025年に3%から12%へと大幅な増加を見せました。しかし、2024年と同様にGzipはオリジンサーバーで使用される最大多数の圧縮アルゴリズムであり続けています（61% Gzip対39% Brotli）。

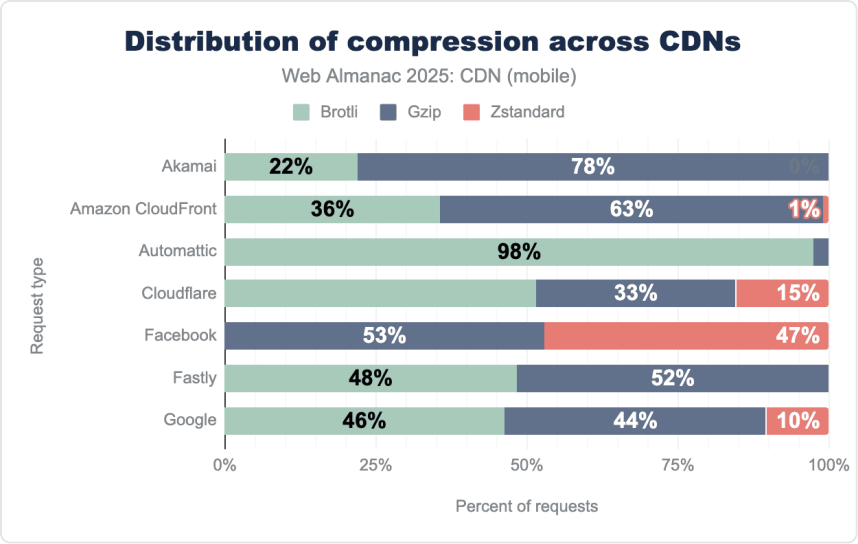


図15.16. CDN全体の圧縮分布（モバイル）。

対象リソースの98%をBrotliで提供しているAutomatticを特に称えます！

採用率では依然として3位ですが、Zstandardは2024年に比べて2025年に進展を見せました。2024年にはFacebookのCDNだけがZstandardの統計的に測定可能な使用を示していたのに対し、2025年にはCloudflare（15%）とGoogle CDN（10%）の両方が測定可能な採用を示しました。Cloudflareの新しい圧縮デフォルトと設定オプションが観察されたデータを説明する可能性があります。データセットの例外はAmazon CloudFrontとAutomatticで、現在ネイティブでZstandard圧縮をサポートしていません。しかし、観察されたデータはCloudFrontがZstandardを使用してオリジンからすでに圧縮されたデータをパススルーしていることを示しました。これは、すべての主要CDNプロバイダーがサポートしていないにもかかわらず、コンテンツオーナーがZstandardを使用したいという要望を示しています。

今後の圧縮トレンド

将来を見据えて、業界は辞書圧縮（Shared Brotli）[479]を探求しており、以下のことが期待されています：

- 繰り返しコンテンツパターンに対して20〜40%優れた圧縮
- 関連リソース間での共有辞書

479. *https://developer.mozilla.org/docs/Web/HTTP/Guides/Compression_dictionary_transport*

- 2024年にChromeオリジントライアルを開始
- 2025〜2026年にCDNサポートの展開が予定

これは圧縮最適化の次のフロンティアを表しており、Brotliの成功を基盤として構築されており、Web Almanacの将来のエディションで注目すべきものです。

TLSの利用

Transport Layer Security（TLS）は安全なウェブ通信の基盤を形成しています。CDNは一貫して新しいTLSバージョンの採用をリードし、インフラのアップグレードを必要とせずにウェブサイトにセキュリティとパフォーマンスの向上をもたらしています。

TLS 1.3の採用

TLS 1.3は、既知の脆弱性を持つ弱い暗号アルゴリズムを含む以前のバージョンと比較して、ウェブトラフィックの全体的なセキュリティを改善します。このプロトコルは最適化された設計によりハンドシェイクレイテンシも削減します。

TLS 1.0と1.1の不在に関する注記: 私たちの測定では、すべてのリクエストにわたってTLS 1.0と1.1の使用がゼロでした。これは、HTTP Archiveが測定に現代のChromeブラウザを使用しており、2020年7月（Chrome 84）にこれらの廃止されたプロトコルのサポートを完全に削除したためです。TLS 1.0または1.1のみをサポートするサーバーは、成功したリクエストではなく接続障害をもたらし、データに表示されません。これらのレガシープロトコルは一部のエンタープライズ環境やIoTデバイスにまだ存在する可能性がありますが、現代のブラウザがアクセスする公開ウェブサービスには使用できなくなっています。

ほぼすべてのCDNトラフィックがTLS 1.3を使用するようになり、99%のリクエストが最新プロトコルバージョンを活用しています。これにより接続確立時間が短縮され、ページの読み込み速度が直接的に改善されます。CDNプロバイダーは新しいウェブ技術の早期採用者であり続けており、CDNを使用するアプリケーションはほとんど追加の労力なしにこれらのパフォーマンスとセキュリティの向上を得ることができます。

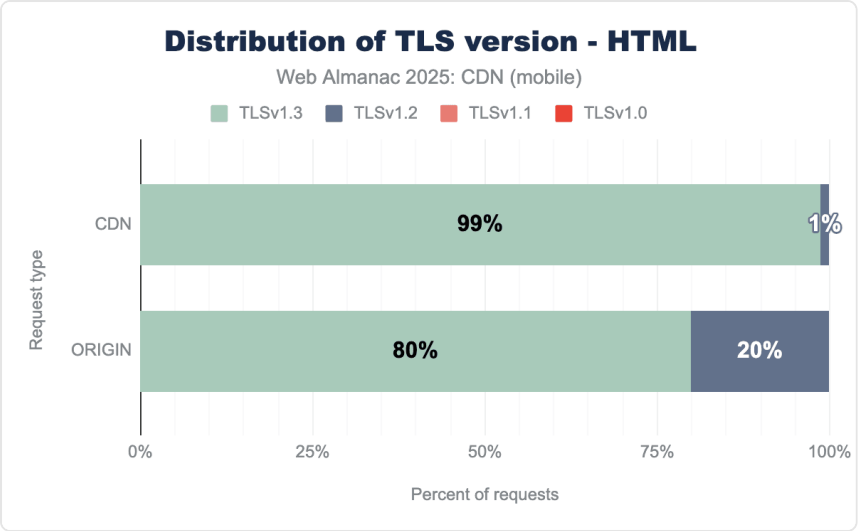


図15.17. HTMLのTLSバージョン分布（モバイル）。

オリジンサーバーのTLS 1.3採用は改善していますが、CDNは依然として自社のソフトウェアとハードウェアのアップグレードを管理する組織と比較して、新しい機能を展開する上での明確な優位性を示しています。

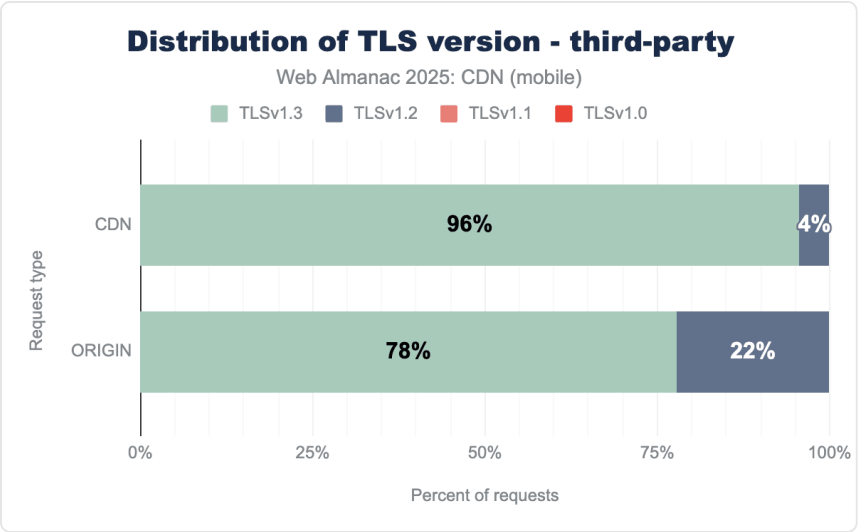


図15.18. サードパーティリクエストのTLSバージョン分布（モバイル）。

サードパーティコンテンツも同様のパターンを示しており、CDNリクエストの96%がTLS 1.3を使用しているのに対し、オリジンサーバーでは78%となっています。CDNの採用率は2024年の93%から2025年の96%へとわずかに増加した一方、オリジンサーバーは同期間に68%から78%へとより大きな増加を示しました。

TLSパフォーマンスへの影響

重要な測定コンテキスト: 以下のTLSネゴシエーション時間は、管理されたデータセンター条件でChromeを使用したHTTP Archiveのシミュレートされたブラウザ接続から測定されています。これらの測定は最適なネットワーク条件とChromeのTLS実装を表しており、クライアント機能、ネットワーク品質、地理的分布の実世界のばらつきを反映しない可能性があります。さらに、ChromeはTLS 1.0/1.1をサポートしなくなったため、これらの測定はTLS 1.2と1.3のパフォーマンス特性のみをキャプチャしています。

TLSネゴシエーション時間は、CDNとオリジンサーバーの間の明確なパフォーマンス差を示しており、デスクトップとモバイル接続の間に追加のばらつきがあります。

デスクトップユーザーはすべてのパフォーマンスレベルで、オリジンサーバーに比べてCDNとのTLSネゴシエーションが大幅に速くなっています。CDNの中央値ネゴシエーション時間は57ミリ秒に対し、オリジンは177ミリ秒と3倍以上速くなっています。このパフォーマンスギャップは分布全体で持続しています。90パーセンタイルでは、CDNが89ミリ秒でネゴシエーションを完了するのに対し、オリジンは277ミリ秒を必要とします。

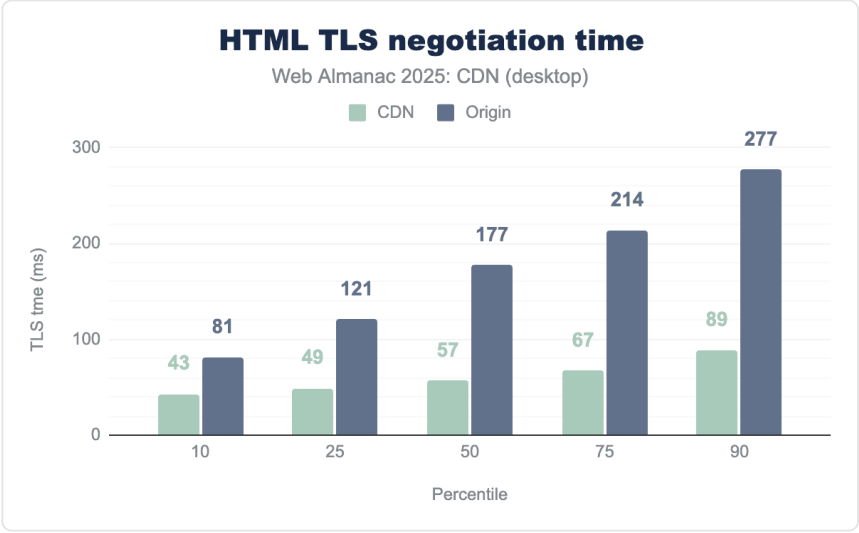


図15.19. HTML TLSネゴシエーション - CDN対オリジン（デスクトップ）。

モバイルデバイスも同様のパターンを示しており、CDNがオリジンサーバーよりも優れたパフォーマンスを発揮していますが、全体のネゴシエーション時間はデスクトップと比較して高くなっています。モバイルCDNの中央値TLSネゴシエーション時間は183ミリ秒、オリジンサーバーは302ミリ秒です。90パーセンタイルでは、モバイルCDNが216ミリ秒かかるのに対し、オリジンサーバーは416ミリ秒を必要とします。

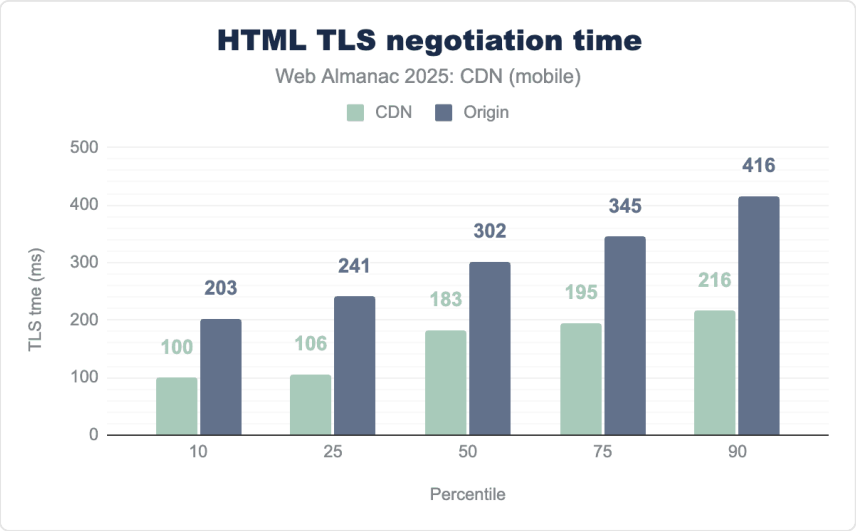


図15.20. HTML TLSネゴシエーション - CDN対オリジン（モバイル）。

モバイルデバイスは一般的に処理能力が限られており、ネットワーク条件が不安定なことが多いため、デスクトップよりも遅いパフォーマンスを示します。CDNがオリジンサーバーよりも優れたパフォーマンスを発揮するのは、主にその分散アーキテクチャのためであり、コンテンツをユーザーに地理的に近い場所に配置し、レイテンシを削減するために接続経路を最適化しています。

画像フォーマットと最適化

画像フォーマットはCDNにおいて引き続き重要な役割を果たし、ウェブサイトのパフォーマンス、帯域幅コスト、全体的なユーザー体験に直接影響を与えています。WebP、AVIF、SVGなどの現代的なフォーマットは、JPEGやPNGなどのレガシーフォーマットと比較して改善された圧縮率と視覚的な忠実度を提供し、最も効率的なオプションであり続けています。これらの効率性は、特にモバイルユーザーや高トラフィックサイトにとって重要な、より速いページ読み込みと低いデータ転送量へと変換されます。

現在、ほとんどのCDNはブラウザの機能を自動的に検出し、利用可能な最も最適化されたフ

ォーマットを提供しています。例えば、ChromeにはAVIF、EdgeにはWebP、レガシーブラウザにはJPEGフォールバックです。アダプティブリサイズ、キャッシング、オンザフライ変換により、デバイスタイプや解像度にわたって静的画像バリアントを維持する必要性は大幅に排除されました。

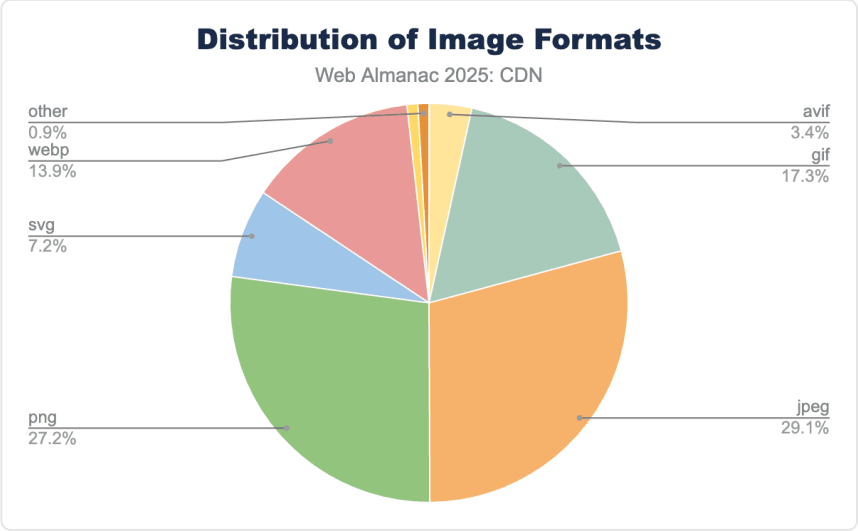


図15.21. 画像フォーマットの分布。

2025年時点で、データは効率性重視のフォーマットへの継続的なシフトを反映しています。JPEGは依然として最もリクエストされるフォーマットですが、2022年と比較して全体で約10%減少し、モバイルではさらに急激な10.5%の減少が見られました。一方、WebP（+5%）、SVG（+2.5%）、AVIF（+3.1%）はすべて2022年以降着実に成長しています。これは業界サポートの逆転ではなく、サイト構成、画像パイプラインのデフォルト、またはフォールバック動作のシフトを反映している可能性があります。

GIFフォーマットは特にモバイルで（デスクトップより+0.5%高く）2.3%の緩やかな増加を示しており、アプリ駆動のウェブトラフィックで一般的なショートループアニメーションとUI要素によって推進されている可能性があります。一方、PNG使用量はわずかな減少（-0.8%）を示しており、デザイナーと開発者がベクターとラスターアセットの両方においてより軽量なフォーマットを優先し続けていることを示しています。

2024年と2025年の間で、AVIF（-1.9%）、SVG（-1.1%）、WebP（-1.8%）でわずかな後退があり、GIF（+2.3%）とJPEG（+2.2%）のわずかな回復で相殺されており、最適化が不十分な配信スタックにおけるフォールバックシナリオや互換性デフォルトの可能性を示しています。

2025年のデータは明確な軌跡を裏付けています：JPEGのようなレガシーフォーマットは依

然としてリクエスト総数を支配していますが、より新しく効率的なフォーマットに徐々にシェアを譲っています。WebP、SVG、AVIFは、レイテンシと帯域幅効率が重要なモバイルファーストのエコシステムで特に、高パフォーマンスコンテンツ配信の新しいベースラインになりつつあります。

クライアントヒント

クライアントヒントはUser-Agent文字列の代替として導入され、ウェブサーバーがHTTPヘッダーを通じてブラウザから特定の情報をリクエストできるようにします。デバイス情報、ユーザーエージェントの設定、ユーザー設定のメディア機能、ネットワーク詳細の4つの主要カテゴリに分類されます。これらのヒントはさらに高エントロピーと低エントロピーのタイプに分かれています。高エントロピーのヒントはフィンガープリンティングに使用される可能性があるため、ブラウザは通常、共有前にユーザーの許可を求めるか他のポリシーを適用します。低エントロピーのヒントはフィンガープリンティングへの有用性が低く、ブラウザまたはユーザーの設定に基づいてデフォルトで送信される場合があります。

2025年も、2024年に確立された同じ方法論を使用してクライアントヒントの測定を継続しました。レスポンスの `Accept-CH` ヘッダーを検出しています。

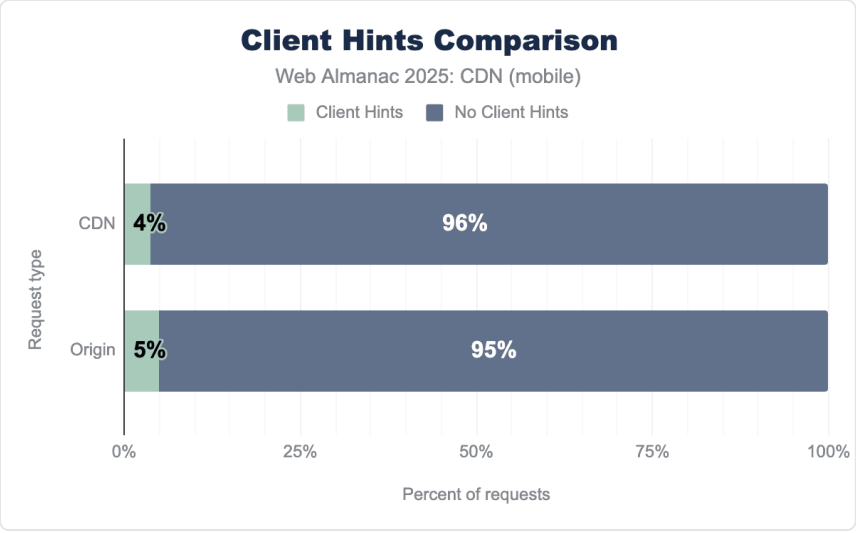


図15.22. クライアントヒントの比較（モバイル）。

クライアントヒントはCDNが採用をリードするという通常のパターンに反しています。CDN使用量は前年比横ばいだった一方、オリジンリクエストは2025年に約5%まで増加しました。可能性の高い要因は、エッジでのキャッシュキーの爆発であり、クライアントヒント

を慎重なバケット管理なしに組み込むとキャッシュヒット率が低下する可能性があります。この課題が、テクノロジーの潜在的な利点にもかかわらず採用率が低いままである理由を説明している可能性があります。

Early Hints

Early Hintsは HTTP 103ステータスコード[480] を使用して、メインレスポンスの準備中にサーバーがブラウザに最初のヘッダーを送信できるようにします。これは、ページ全体の準備が整う前にスタイルシート、JavaScriptファイル、フォントなどの重要なリソースをプリロードするのに特に役立ちます。パフォーマンス向上の可能性があるにもかかわらず、採用は依然として最小限にとどまっています。

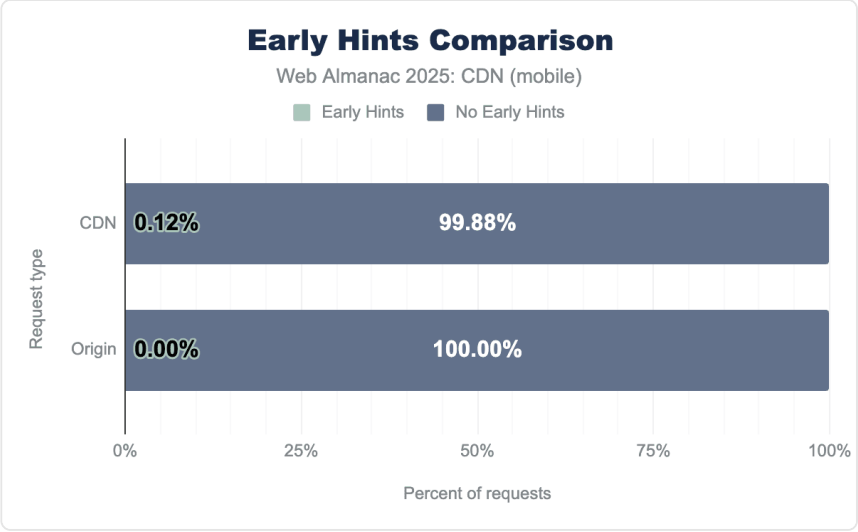


図15.23. Early Hintsの比較（モバイル）。

Early Hintsへのブラウザサポートは広まっていますが、データセットではCDNリクエストのわずか0.012%という最小限の使用しか見られませんでした。これは2024年から2025年にかけてわずか0.002%の増加を示しています。Vercelは1%以上の採用（2.84%）をサポートした唯一のCDNで、CloudflareとFastlyは1%未満でした。

より多くのサイトがEarly Hintsを使用し始めるにつれて、パフォーマンスにどのような影響を与えるかを見ることに関心があります。来年のWeb Almanacまでに、より多くのCDNプロバイダーがこの機能を実装し、その影響に関する詳細な統計を共有できるだけのデータが集

480. https://datatracker.ietf.org/doc/html/rfc8297#section-2

まることを期待しています。

結論

2024年にはCDNがHTTP/3などの新興技術の採用で先頭に立ち、そのパターンは2025年にも続いています。BrotliやZstandard圧縮、TLS 1.3暗号化などの機能を見ると、CDNはサーバー、ロードバランサー、ネットワーク機器の全フリートを刷新するのではなく、シンプルな設定変更でサイトがこれらの改善を実装しやすくしています。

2025年の分析では、CDNがウェブプロトコル進化の先陣を切っていることが明らかになりました。Cloudflareなどの主要プロバイダーはHTTP/3採用率69%を達成しているのに対し、オリジンサーバーは5%未満にとどまっています。この10〜15倍の差は、エッジインフラが次世代ウェブ標準の採用をどのように推進しているかを示しています。

2025年分析の主な発見：

- **プロトコルリーダーシップ**: CDNはHTTP/3採用率29〜45%に対し、オリジンは7%未満
- **セキュリティの優秀性**: CDNトラフィックの95.6%がTLS 1.3を使用、オリジンサーバーの77.7%と比較
- **圧縮革新**: Automatticなどのリーダーは97.5%のBrotli採用を達成し、Zstandardは前年比3%から12%へ成長
- **パフォーマンス優位性**: CDNはモバイルで19%高速、デスクトップでは60%高速なDNSレスポンスを提供

今年はHTTP/3をより深く調査し、2024年に初めて検討したEarly Hintsを再訪しました。初めてCDNのパフォーマンスとセキュリティを切り分け、2026年には両トピック間のトレードオフについてより深く掘り下げる予定です。当初はIPv6分析を含める予定でしたが、データが有意な結論を引き出すのに十分な信頼性を持っていませんでした。より堅牢な測定ができ次第、2026年のチャプターでIPv6の採用に取り組む予定です。

2025年のCDN環境は、これらのプラットフォームが単純なコンテンツ配信を大幅に超えて進化し、現代ウェブの必須インフラである包括的な最適化およびセキュリティプラットフォームとなっていることを示しています。

このチャプターのいくつかのトピックが拡張され、異なる視点からデータを提供している2025年版Web AlmanacのSecurityチャプターも読者に訪問することをお勧めします。

2026年も再びご参加ください。さらなるデータを収集・分析して、読者の皆さんと新しい

洞察を共有できることを楽しみにしています。

著者

Joe Viggiano

joeviggiano

Joe ViggianoはAmazon Web Servicesのプリンシパルソリューションアーキテクトで、メディア＆エンターテインメント顧客が大規模にメディアコンテンツを配信できるよう支援しています。

Alex Moening

AlexMoening

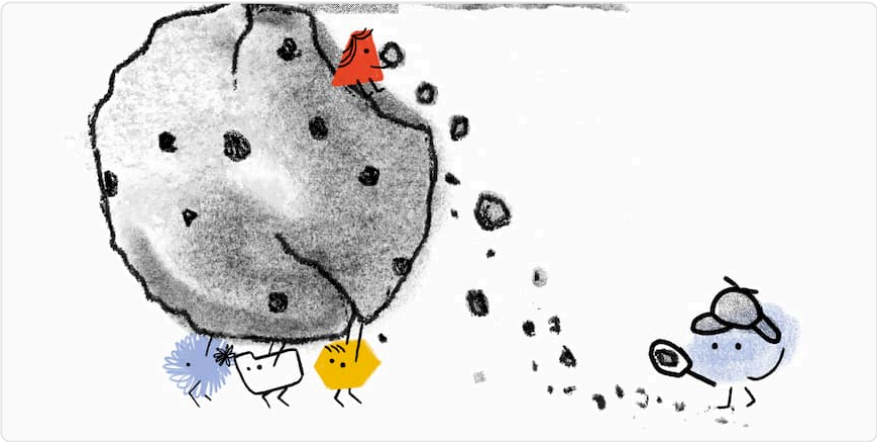
Alex MoeningはAmazon Web Servicesのシニアエッジソリューションアーキテクトです。

Nick McCord

nick-mccord nicholasmccord

Nick McCordはAmazon Web Servicesのソリューションアーキテクトで、スタートアップとエッジサービスに注力しています。

部 IV 章 16
Cookie



***Yohan Beugin** によって書かれた。*
Jannis Rautenstrauch と Martina Kraus によってレビュ。
Chris Böttger による分析。
Brian Smith と Barry Pollard 編集。
Sakae Kotaro によって翻訳された。

はじめに

Cookie[481]は、ステートレスなプロトコルであるHTTPリクエストにわたって、ウェブサイトがデータを保存し状態情報を維持できるようにします。ウェブアプリケーションは認証、不正防止とセキュリティ、設定やユーザーの選択の記憶など、さまざまな目的でCookieを使用します。しかし、1990年代中頃に登場して以来、Cookieはウェブユーザーのオンライントラッキングでも支配的な役割を果たしてきました。

長年にわたり、Brave、Firefox、Safariなどのブラウザベンダーはサードパーティ Cookieに対して制限を課し、パーティション化し、削除[482]してきました。Chromeも当初は同様の手順に従うように見え、すべてのサードパーティ Cookieをブロックする計画[483]を発表しましたが、複数の延期の後、Googleは最終的にサードパーティ Cookieの制限を維持せず、ユーザー

481. https://developer.mozilla.org/docs/Web/HTTP/Cookies
482. https://developer.mozilla.org/docs/Web/Privacy/Guides/Third-party_cookies#how_do_browsers_handle_third-party_cookies
483. https://blog.chromium.org/2020/01/building-more-private-web-path-towards.html

がChromeで無効にするかどうかを決定できるようにする[484]ことを決定しました。

本チャプターでは、2025年7月のHTTP Archiveクロールによって訪問されたウェブページで遭遇したウェブCookieの普及状況と構造を測定し報告します。特に言及がない限り、これらの結果の大半は、クロール時にHTTP Archiveデータセットに記録されたChrome User Experience report（CrUX）のランクに基づく上位100万（トップミリオン）の人気ウェブサイトに対するものです。デスクトップとモバイルデバイスの両方の結果も示していますが、実際のところ本チャプターでは2種類のデバイス間に有意な差はほとんど見られません。

背景

まず、本チャプターで使用されるいくつかの用語について共通の理解を得ましょう。

HTTP Cookie

ユーザーがウェブサイトを訪問すると、ユーザーのウェブブラウザにHTTP Cookie[485]を設定・保存するよう要求できるウェブサーバーとやり取りします。このCookieはユーザーのデバイスにテキスト文字列として保存されたデータに対応し、以降のHTTPリクエストとともにウェブサーバーに送信されます。Cookieは複数のHTTPリクエストにわたってユーザーのステートフルな情報を維持するために使用されます。これにより認証、セッション管理、トラッキングが可能になります。また、CookieはプライバシーとセキュリティリスクとIも関連しています。

ファーストパーティCookieとサードパーティCookie

Cookieはウェブサーバーによって設定され、ファーストパーティとサードパーティの2種類があります。ファーストパーティCookieはユーザーが訪問しているサイトと同じドメインによって設定され、サードパーティCookieは異なるドメインから設定されます。

サードパーティCookieはサードパーティからのもの、またはトップレベルサイトと同じ「ファーストパーティ」に属する異なるサイトやサービスからのものである場合があります。サードパーティCookieは実質的にクロスサイトCookieです。

例えば、`example.com`のドメインのオーナーが`example.net`も所有しており、`https://www.example.com`を訪問するユーザーに対して以下のCookieが設定されているとします：

484. *https://privacysandbox.com/news/update-on-plans-for-privacy-sandbox-technologies/*
485. *https://developer.mozilla.org/docs/Web/HTTP/Cookies*

Cookie名	設定元	Cookieの種類	理由
cookie_a	www.example.com	ファーストパーティ	訪問したウェブサイトと同じドメイン
cookie_b	cart.example.com	ファーストパーティ	訪問したウェブサイトと同じドメイン：サブドメインは関係しない
cookie_c	www.example.edu	サードパーティ	訪問したウェブサイトと異なるドメイン
cookie_d	tracking.example.org	サードパーティ	訪問したウェブサイトと異なるドメイン
cookie_e	login.example.net	サードパーティ	この例では同じオーナーが所有していても訪問したウェブサイトとは異なるドメイン（トップレベルサイトの同じ「ファーストパーティ」からのクロスサイトCookie）

図16.1. Cookieのコンテキスト。

プライバシーとセキュリティのリスク

Cookieはウェブの機能に欠かせませんが、プライバシーとセキュリティのリスクをもたらします：

- **ウェブトラッキング。** Cookieはサードパーティによってウェブサイトをまたいでユーザーを追跡し、ブラウジング行動と興味を記録するために使用されます。ターゲット広告では、このデータがユーザーの興味に合わせた広告を表示するために活用されます。

  このトラッキングは通常次のように行われます：サイトに埋め込まれたサードパーティコードがユーザーを識別するCookieを設定できます。次に、同じサードパーティは、同じく埋め込まれている他のウェブサイトをユーザーが訪問したときにそのCookieを取得することでユーザーアクティビティを記録できます（Privacyチャプターも参照）。

  ファーストパーティCookieもオントラッキングに使用できることに注意が必要です。Cookie同期などの方法によりサードパーティCookieの制限を回避し、

ユーザーを異なるウェブサイトにわたって[486]追跡することが可能です。

- **Cookieの盗難とセッションハイジャック。** Cookieは複数のHTTPリクエストにわたる認証のために、認証情報（例えばセッショントークン）などのセッション情報を保存するために使用されます。ただし、これらのCookieが悪意のある攻撃者に入手された場合、対応するウェブサーバーへの認証に使用される可能性があります。

  Cookieがウェブサーバーによって適切に設定されていない場合、セッションハイジャック[487]、クロスサイトリクエストフォージェリ（CSRF）[488]、クロスサイトスクリプトインクルージョン（XSS）[489]などのクロスサイト脆弱性にさらされる可能性があります（Securityチャプターも参照）。

ファーストパーティとサードパーティの普及率

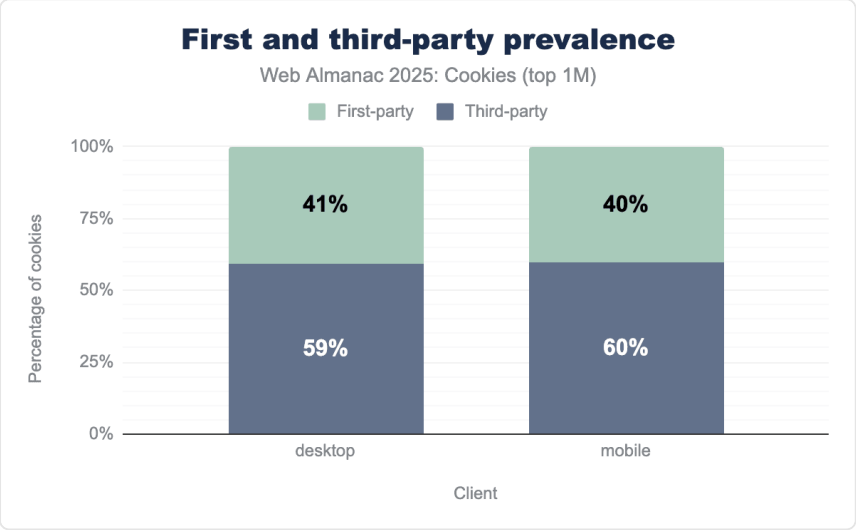


図16.2. ファーストパーティとサードパーティの普及率。

2025年7月のHTTP Archiveクロールによる上位100万の人気ウェブサイトにおけるファース

486. *https://dl.acm.org/doi/abs/10.1145/3442381.3449837*
487. *https://developer.mozilla.org/docs/Glossary/Session_Hijacking*
488. *https://developer.mozilla.org/docs/Web/Security/Practical_implementation_guides/CSRF_prevention*
489. *https://developer.mozilla.org/docs/Glossary/Cross-site_scripting*

トパーティとサードパーティCookieの全体的な普及率は、昨年の分布[490]と同様です。

デスクトップとモバイルデバイスの両方で、Cookieの約40%がファーストパーティ、約60%がサードパーティです。

ランク別のファーストパーティとサードパーティの普及率

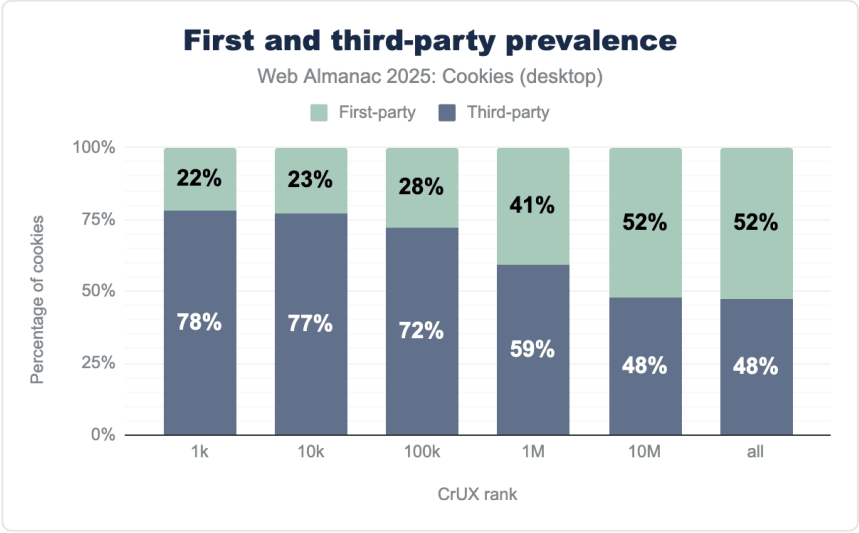


図16.3. デスクトップにおけるランク別のファーストパーティとサードパーティCookieの普及率。

490. https://almanac.httparchive.org/ja/2024/cookies#ファーストパーティとサードパーティの普及率

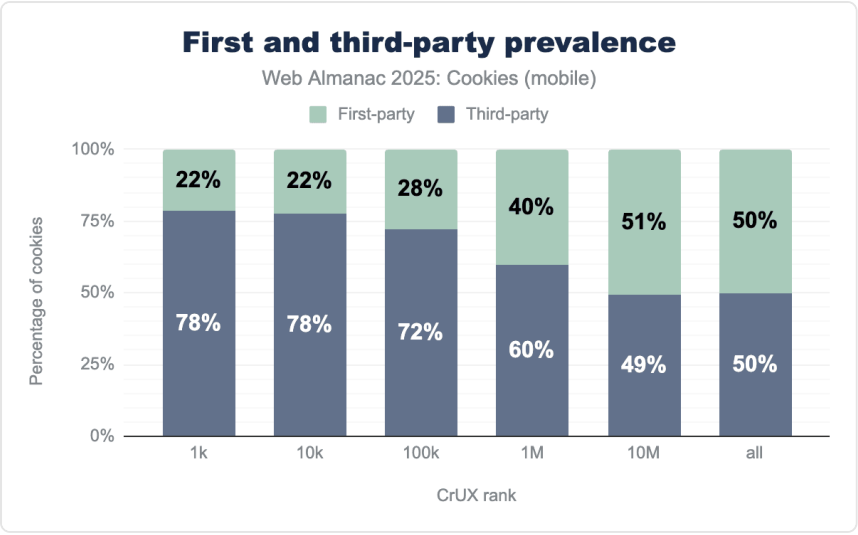


図16.4. モバイルにおけるランク別のファーストパーティとサードパーティCookieの普及率。

最も人気のあるウェブサイトが比例してファーストパーティよりもサードパーティCookieを多く設定していることが観察されます：最も訪問される上位1,000サイトではCookieの78%がサードパーティですが、上位1,000万サイトでは50%をわずかに下回ります。これは、より人気のあるウェブサイトもより多くのサードパーティコンテンツとスクリプトを含んでおり、それらがさまざまな機能を有効にするためにサードパーティCookieを設定するという事実によって説明される可能性があります。

Cookieの属性

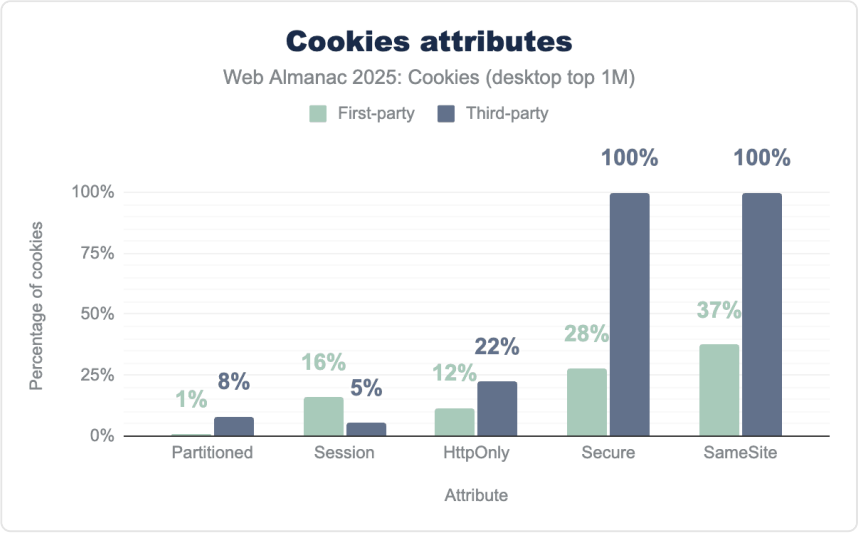


図16.5. デスクトップにおけるCookie属性の概要。

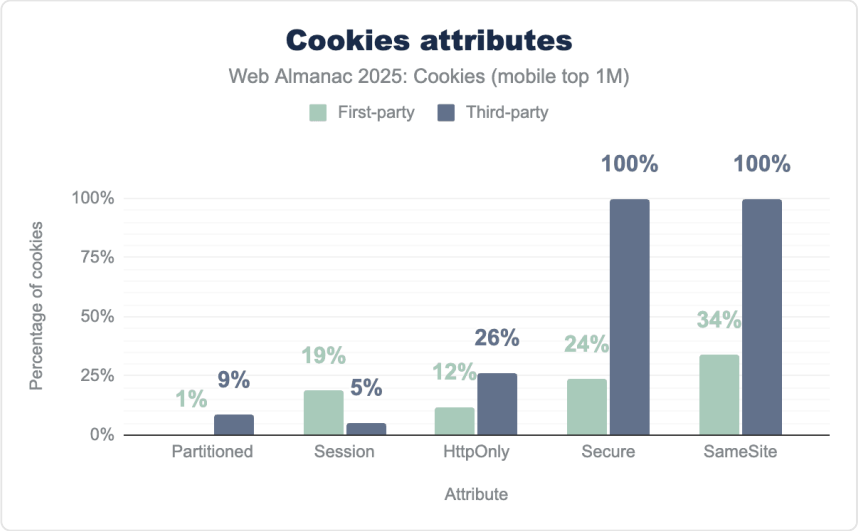


図16.6. モバイルにおけるCookie属性の概要。

データは観察された各種Cookieのさまざまな Cookie属性[491]を示しています。それぞれについ

491. https://developer.mozilla.org/docs/Web/HTTP/Headers/Set-Cookie

て詳しく見ていきましょう。

`Partitioned`（CHIPSプロポーザル）

対応ブラウザ[492]では、パーティション化されたCookieは、トップレベルサイトごとにパーティション化されたストレージに配置することで、サードパーティCookieがクロスサイトトラッキングに使用されるのを防ぎます。

8.6%

図16.7. モバイルページでの `Partition` Cookieの割合。

2025年7月、上位100万のサードパーティCookieの約9%がパーティション化されています。これは昨年の結果の6%[493]と比較して、この比較的新しい属性の採用がわずかに増加したことを示しています。

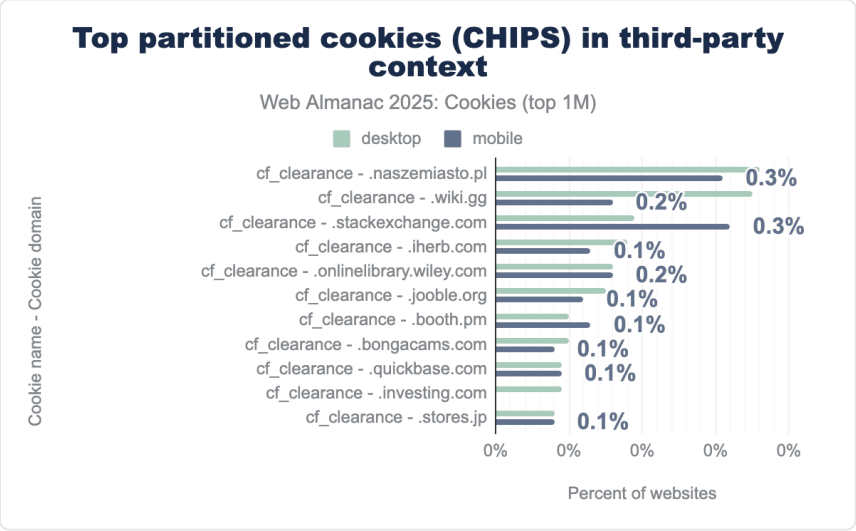


図16.8. サードパーティコンテキストでのトップパーティション化Cookie（CHIPS）。

上記のグラフは、2025年7月のウェブページのサードパーティコンテキストで見つかった最も一般的な10のパーティション化Cookie（名前とドメイン）を示しています。ここで、昨年

492. https://developer.mozilla.org/docs/Web/Privacy/Privacy_sandbox/Partitioned_cookies#browser_compatibility
493. https://almanac.httparchive.org/ja/2024/cookies#partitioned

の分析からの大きな変化が観察されます。実際、2025年のサードパーティパーティション化Cookieの全体的な使用量は非常に低いレベルに急落したようです。

興味深いことに、2024年にある程度主流だったパーティション化Cookie（パーティション化Cookieを持つウェブサイトの約9%）はもはや存在しません。これらのCookieのうち2つはYouTubeによって設定されたものであり、もう1つはChromesのPrivacy Sandboxテストフェーズに参加[494]したドメインによって設定された `receive-cookie-deprecation` Cookieでした。代わりに、2025年の最も一般的なパーティション化サードパーティCookieTOP10の全体をCloudflareの `cf_clearance` Cookieが占めています。

したがって、過去1年間でYouTubeは `youtube.com` と他のウェブサイトに埋め込まれたビデオiframeでのこれらのCookieの設定方法を変更したようです。これらの変更を説明できる潜在的な理由には、誤った設定、A/Bテスト、そしてより可能性が高いのは、パーティション化Cookie（CHIPSプロポーザル）のサポートが継続されているにもかかわらず、Privacy Sandbox APIの一時停止とその後の廃止に関するGoogleの発表に続くインフラやポリシーの更新が含まれます。

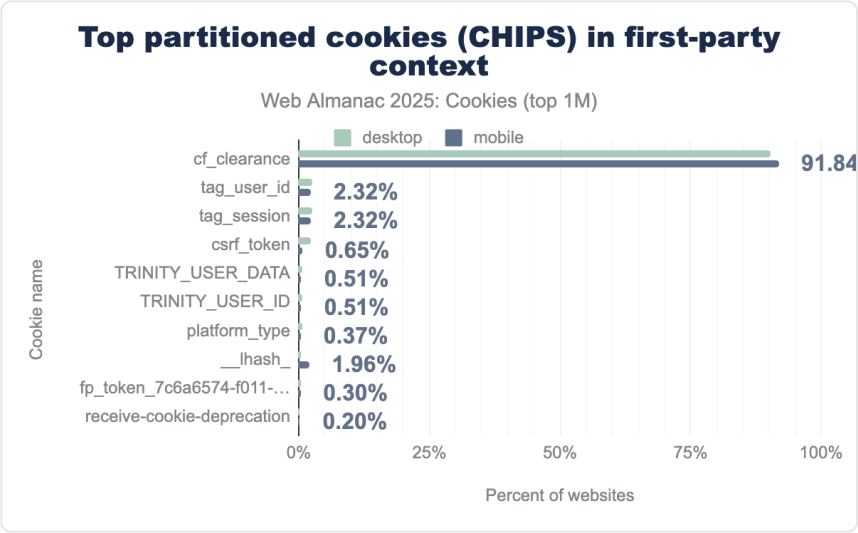


図16.9. ファーストパーティコンテキストでのトップパーティション化Cookie（CHIPS）。

2025年においても、ファーストパーティCookieの1%がパーティション化として設定されていることが引き続き観察されます。これはやや意外かもしれません。CHIPSプロポーザルは主にサードパーティCookieのパーティション化に関するものであり、パーティション化されたファーストパーティCookieの特定のまれなケース[495]を述べていても、動作要件[496]はファー

494. https://developers.google.com/privacy-sandbox/private-advertising/setup/web/chrome-facilitated-testing
495. https://github.com/privacycg/CHIPS?tab=readme-ov-file#first-party-chips
496. https://github.com/privacycg/CHIPS/issues/51

ストパーティコンテキストでは不明確です。一つの理由として、一部のCookieが常に同じ方法で設定されており、設定するウェブサーバーが現在ファーストパーティかサードパーティかを区別していない可能性があります。

2025年、これらのファーストパーティパーティション化Cookieの90%以上がボット検出に関連するCloudflareの `cf_clearance` Cookieです。2024年の分析と比較すると、Privacy Sandbox APIテストに参加しているドメインによって設定されたファーストパーティパーティション化Cookie `receive-cookie-deprecation` は、もはやそれほど人気がありません。おそらく、この観察は過去1年間のGoogleの対応する発表によるこれらのAPIの採用の一時停止または減少によって説明できるかもしれません。

セッション

ファーストパーティCookieの19%とサードパーティCookieの7%はセッションCookieです。これらは単一のユーザーセッションにのみ有効な一時的なCookieで、ユーザーが設定された対応するウェブサイトを終了するか、ウェブブラウザを閉じるか、どちらか早い方が起きると期限切れになります。

`HttpOnly`

`HttpOnly` Cookieは、JavaScriptコードからアクセスできないため（ただし、JavaScriptから開始された `XMLHttpRequest` や `fetch` リクエストとともに送信されます）、クロスサイトスクリプティング（XSS）[497]に対するある程度の軽減策を提供します。

ファーストパーティCookieの12%とサードパーティCookieの26%強がこの属性を設定しています。

`Secure`

`Secure` CookieはHTTPSを通じて行われたリクエストにのみ送信されます。昨年の傾向と同様[498]で、ファーストパーティCookieの24%のみがこの属性を設定しているのに対し、すべてのサードパーティCookieは `SameSite=None` を使用したい場合はこの属性を設定する必要があります（以下を参照）。

497. *https://developer.mozilla.org/docs/Glossary/Cross-site_scripting*
498. *https://almanac.httparchive.org/ja/2024/cookies#secure*

`SameSite`

`SameSite` Cookie属性は、サイトがクロスサイトリクエストにCookieを含めるタイミングを指定できるようにします：

- `SameSite=Strict` ：CookieはCookieのオリジンと同じサイトからのリクエストに応じてのみ送信されます。
- `SameSite=Lax` ：ブラウザがCookieのオリジンサイトへのナビゲーション時にもCookieを送信する点を除いて `SameSite=Strict` と同じです。Chromeでは、値が設定されていない場合のデフォルト値が `SameSite=Lax` です。
- `SameSite=None` ：CookieはサイトとCookieの同一サイトまたはクロスサイトリクエストで送信されます。 これはCookieによるサードパーティトラッキングを可能にするために、トラッキングCookieは `SameSite` 属性を `None` に設定する必要があることを意味します。

`SameSite` 属性の詳細については、以下の参考資料を参照してください：

- `SameSite` cookies explained
- “Same-site” and “same-origin”[499]
- What are the parts of a URL?[500]

499. *https://web.dev/articles/same-site-same-origin*
500. *https://web.dev/articles/url-parts*

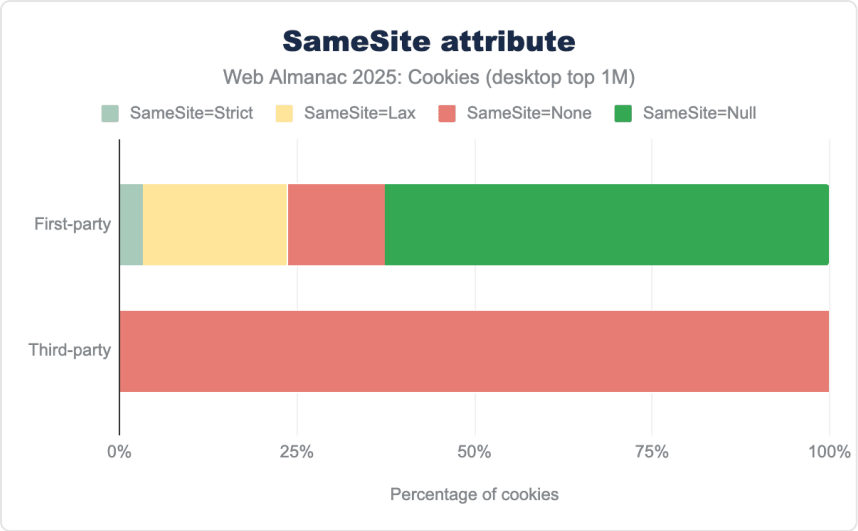


図16.10. デスクトップにおけるCookieの `SameSite` 属性。

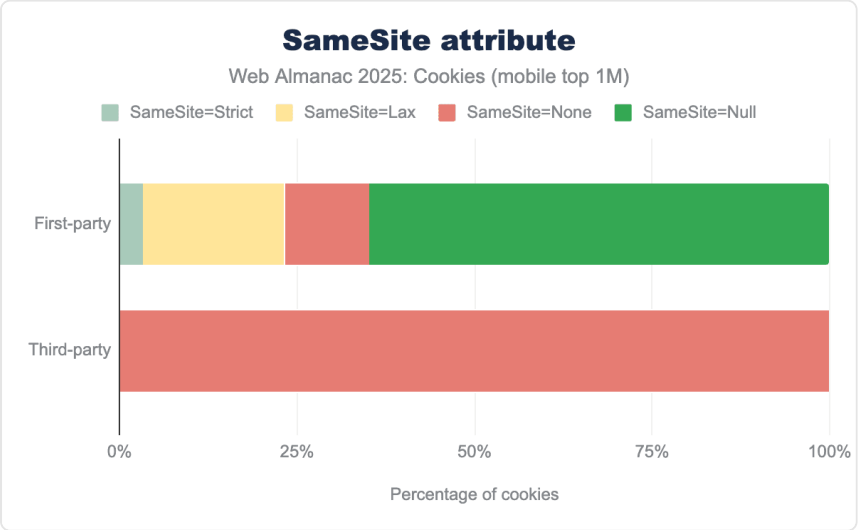


図16.11. モバイルにおけるCookieの `SameSite` 属性。

クライアントにわたるファーストパーティとサードパーティCookieのこの属性の全体的な分布は昨年のもの[501]と同様です。サードパーティCookieのほぼ100%がクロスサイトリクエストで送信され（`SameSite=None`）、これによりクロスサイトトラッキングが可能になりま

501. *https://almanac.httparchive.org/ja/2024/cookies#samesite*

す。

ファーストパーティCookieの大半（デスクトップで66%、モバイルで62%）はこの属性を設定しておらず、Chromeはデフォルトの`Lax`動作を割り当てます。これはファーストパーティCookieの他の19%が明示的に選択している動作と同じです。`Strict`設定は3%のみで、残りの11%はサイトとCookieの同一サイトとクロスサイトの両方のリクエストで送信されています（`SameSite=None`）。

Cookieのプレフィックス

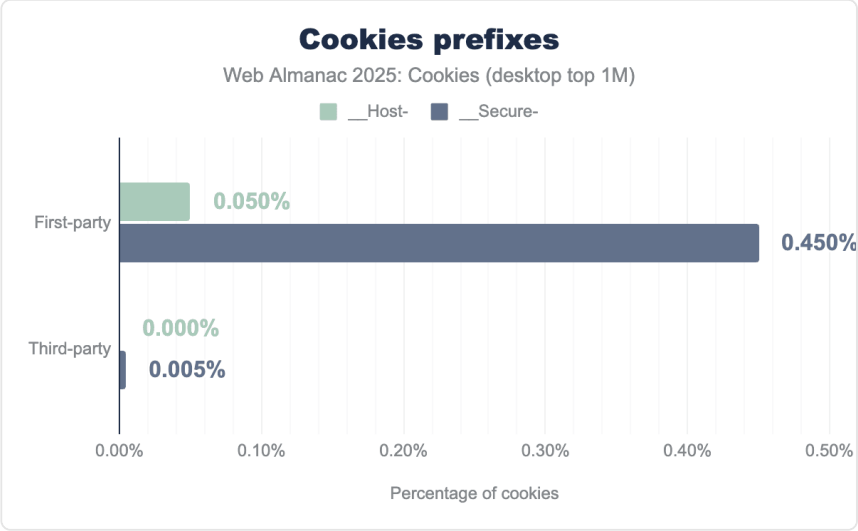


図16.12. デスクトップページで観察されたCookieのプレフィックス。

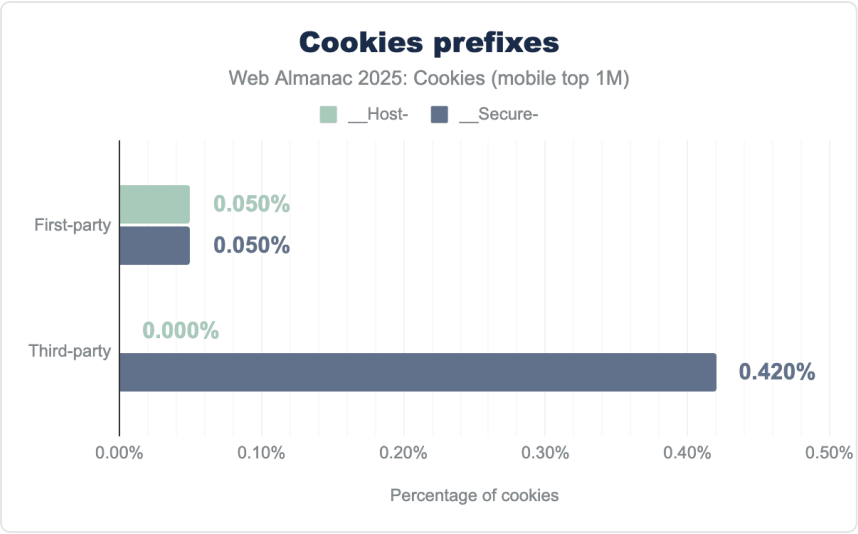


図16.13. モバイルページで観察されたCookieのプレフィックス。

Cookieのプレフィックス[502] `__Host-` と `__Secure-` はCookie名に使用して、安全なHTTPSオリジンによってのみ設定または変更できることを示すことができます。これはセッション固定[503]攻撃から防御するためのものです。

両方のプレフィックスを持つCookieは安全なHTTPSオリジンによって設定され、`Secure` 属性が設定されている必要があります。さらに、`__Host-` Cookieは `Domain` 属性を含まず、`Path` を `/` に設定する必要があります。そのため、`__Host-` Cookieは設定されたままさにそのホストにのみ送り返され、親ドメインには送られません。

ここでは昨年[504]と同じ結論を導きます。これらのプレフィックスは10年前に導入されて以来、ウェブでの採用率が非常に低く、実際には提供する多層防御の措置は使用されていません。

502. *https://developer.mozilla.org/docs/Web/HTTP/Cookies#cookie_prefixes*
503. *https://developer.mozilla.org/docs/Web/Security/Types_of_attacks#session_fixation*
504. *https://almanac.httparchive.org/ja/2024/cookies#cookieのプレフィックス*

上位のCookieとそれらを設定するドメイン

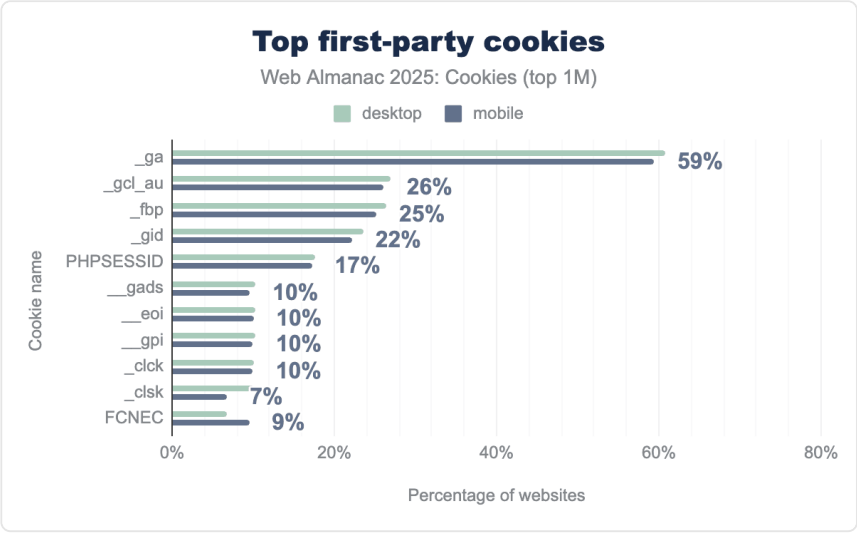


図16.14. 設定されたトップファーストパーティCookie。

上記のグラフは設定されている最も一般的なファーストパーティCookie名トップ10を報告しています。 Google Analyticsはウェブサイトの統計、分析レポート、ターゲット広告に使用される `_ga` と `_gcl_au` Cookieをウェブサイトの約60%と25%で設定しています。 このトップ10に存在する他のCookieはオンライントラッキング、ユーザーのセッションを識別するために使用されるセッションCookie、またはパフォーマンスに関連しています。

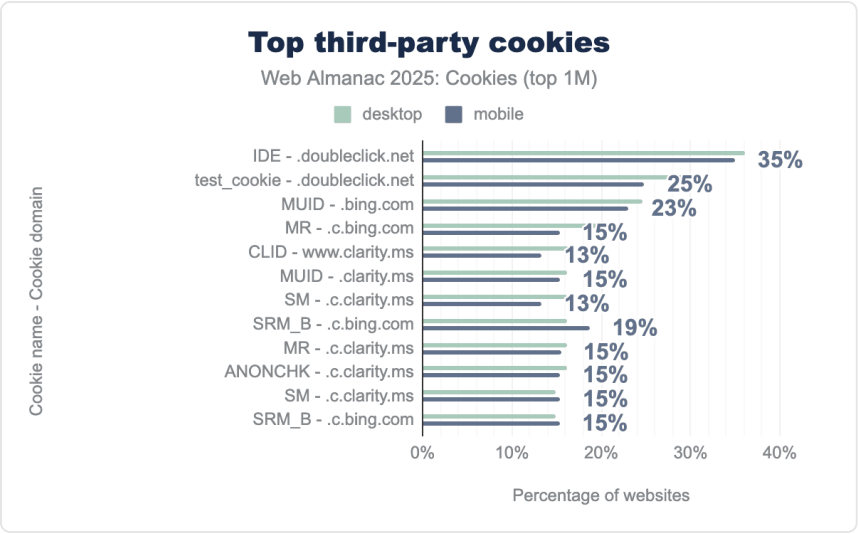


図16.15. トップサードパーティCookieとそれらを設定するドメイン。

同様に、この図は上位100万のウェブサイトで作成されている最も一般的なサードパーティCookieトップ10を示しています。

`IDE`と`test_cookie` Cookieは`doubleclick.net`（Googleが所有）によって設定され、それぞれウェブサイトの35%以上と25%以上に存在します。DoubleClickは`test_cookie`を設定しようとすることで、ユーザーのウェブブラウザがサードパーティCookieをサポートしているかどうかを確認します。

Microsoftの`MUID`が次に来て、ウェブサイトの23%以上に存在し、ターゲット広告とクロスサイトトラッキングにも使用されています。

`Partitioned` Cookieセクションですでに指摘したように、今年はトップサードパーティCookieの中にYouTubeの`YSC`と`VISITOR_INFO1_LIVE`が見られなくなりました。これは2024年の分析以降、YouTubeの変更（おそらくこちら[505]のようなPrivacy Sandboxプロポーザルに関するGoogleの発表と関連している）によるものと思われます。埋め込みページが読み込まれただけでビデオが再生されていない場合、これらのCookieがもはや設定されていないようです。さらに、GoogleのPrivacy & Terms[506]も`VISITOR_INFO1_LIVE`が`__Secure-YNID` Cookieに置き換えられていることを文書化しています。

505. *https://privacysandbox.google.com/blog/privacy-sandbox-next-steps*
506. *https://policies.google.com/technologies/cookies?hl=en-US*

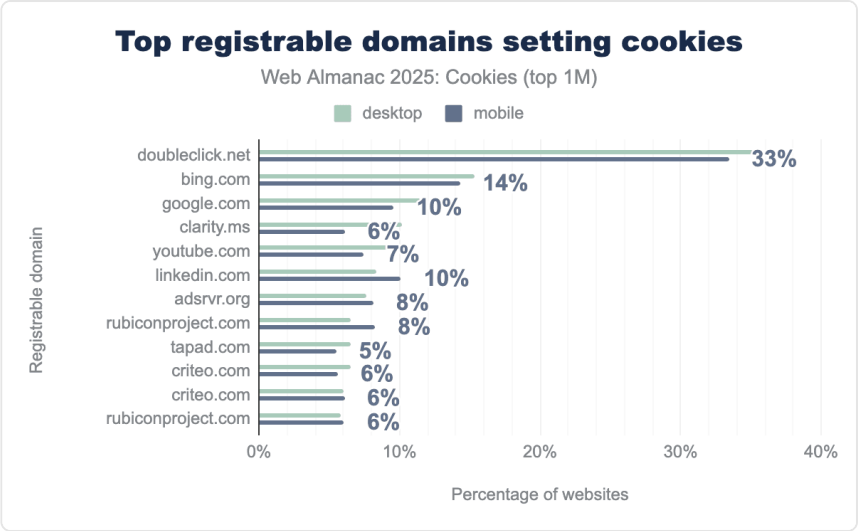


図16.16. Cookieを設定するトップ登録可能ドメイン。

以前の結果から予想通り、ウェブでCookieを設定する最も一般的な10のドメインはすべて、検索、ターゲティング、広告サービスに関わっています。

Googleのカバレッジ（`doubleclick.net`、`google.com`、`youtube.com`）はウェブサイトの少なくとも33%に達しており、Microsoftの（`bing.com`、`clarity.ms`、`linkedin.com`）は少なくとも14%です。

ウェブサイトによって設定されるCookieの数

パーセンタイル	ファーストパーティ	サードパーティ	全て
min	1	1	1
p25	3	2	4
median	7	7	9
p75	13	16	23
p90	22	40	44
p99	45	399	395
max	178	885	915

図16.17. 上位100万デスクトップページで設定されたCookieの数に関する統計。

パーセンタイル	ファーストパーティ	サードパーティ	全て
min	1	1	1
p25	3	2	4
median	6	4	9
p75	12	15	22
p90	21	39	43
p99	45	400	396
max	178	801	831

図16.18. 上位100万モバイルページで設定されたCookieの数に関する統計。

ウェブサイトは全体的にCookieの中央値として9個を設定しており、デスクトップではファーストパーティ7個とサードパーティ7個、モバイルではファーストパーティ6個とサードパーティ4個です。

表はウェブサイトごとに観察されたCookieの数に関するその他のいくつかの統計を報告しており、下のグラフはその累積分布関数（cdf）を示しています。例えば、デスクトップではウェブサイトごとに最大178個のファーストパーティと885個のサードパーティCookieが設定されます：

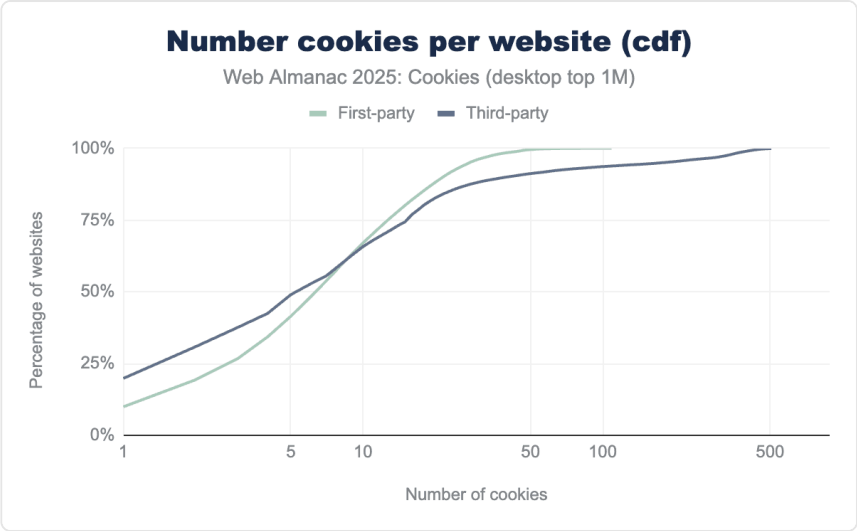


図16.19. デスクトップページにおけるウェブサイトごとのCookieの数（cdf）。

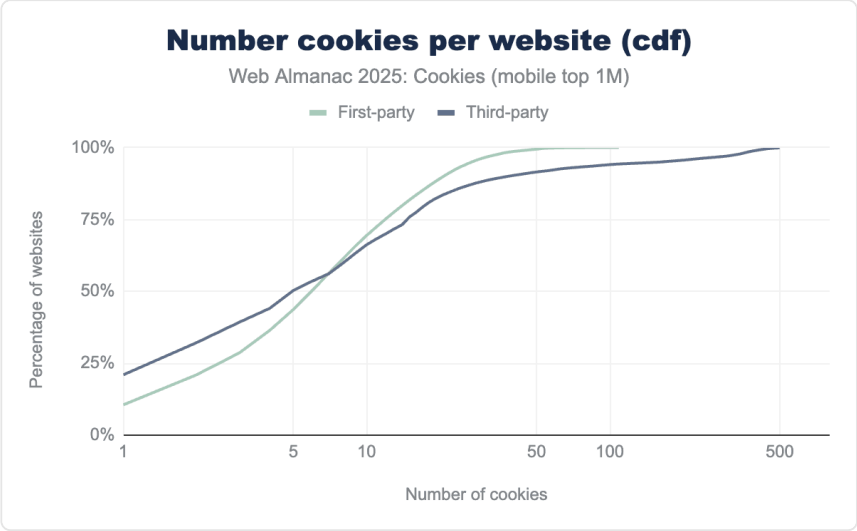


図16.20. モバイルページにおけるウェブサイトごとのCookieの数（cdf）。

Cookieのサイズ

パーセンタイル	ファーストパーティ	サードパーティ	全て
min	1	1	1
p25	29	22	24
median	41	39	40
p75	67	59	64
p90	157	145	149
p99	414	321	338
max	4090	4096	4096

図16.21. 上位100万デスクトップページで設定されたCookieのサイズに関する統計。

パーセンタイル	ファーストパーティ	サードパーティ	全て
min	1	1	1
p25	22	29	24
median	39	41	40
p75	62	67	65
p90	145	162	150
p99	326	414	388
max	4096	4081	4096

図16.22. 上位100万モバイルページで設定されたCookieのサイズに関する統計。

観察されたすべてのCookieにわたるCookieのサイズの中央値は40バイトで、最大4KBであり、これはRFC 6265[507]で定義された制限と一致しています。

昨年[508]と同様に、1バイトのサイズのCookieが観察されます。これらはおそらく空の`Set-Cookie`ヘッダーによってエラーで設定されたものです。

各クライアントの上位100万サイトで見られたすべてのCookieのサイズの累積分布関数（cdf）をグラフ化できます：

507. *https://datatracker.ietf.org/doc/html/rfc6265#section-6.1*
508. *https://almanac.httparchive.org/ja/2024/cookies#cookieのサイズ*

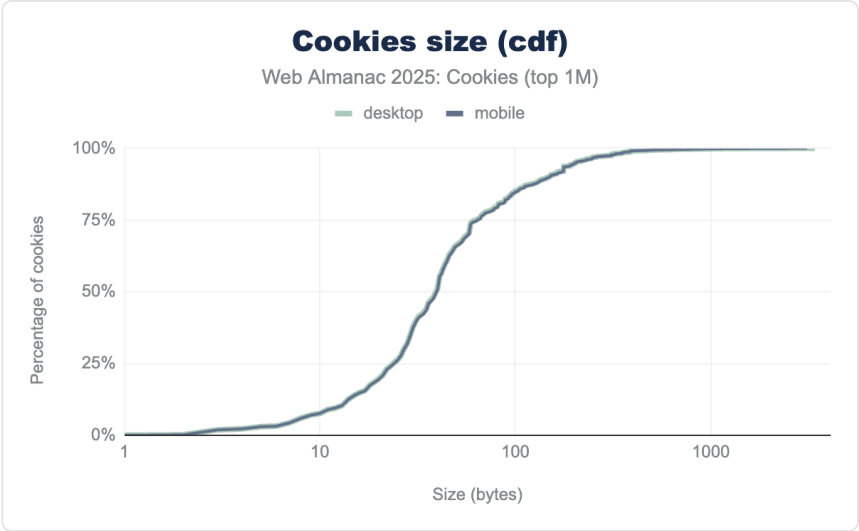


図16.23. デスクトップとモバイルページにおけるウェブサイトごとのCookieのサイズ（cdf）。

永続性（有効期限）

パーセンタイル	ファーストパーティ	サードパーティ	全て
min	0	0	0
p25	1	30	21
median	365	360	364
p75	395	365	390
p90	400	400	400
p99	400	400	400
max	400	400	400

図16.24. 上位100万デスクトップページで設定されたCookieの有効期間に関する統計。

パーセンタイル	ファーストパーティ	サードパーティ	全て
min	0	0	0
p25	1	30	30
median	365	270	360
p75	395	365	390
p90	400	400	400
p99	400	400	400
max	400	400	400

図16.25. 上位100万モバイルページで設定されたCookieの有効期間に関する統計。

Cookieは作成時に有効期限が設定されます。セッションCookieがセッション終了直後に期限切れになる（前のセクションを参照）のに対し、ほとんどのファーストパーティとサードパーティCookieはそうではなく、中央値の有効期間が丸1年となっています。

Cookieの存続期間が長いほど、再識別やクロスサイトトラッキングに使用できる期間が長くなります。これが、ほとんどのトラッキングCookieが通常ブラウザに長期間保存されるよう設定される理由です。

本チャプターのHTTP Archiveツールの計測・収集で観察できるCookieの最大有効期間は400日です。これはChromeがCookieの`Expires`と`Max-Age`属性に課すハードリミット[509]によるものです。

結論

本チャプターの観察は昨年の分析の結論[510]を確認するものです：

- ウェブで遭遇するCookieの大半（60%）はサードパーティCookieであり、人気サイトはそれほど人気でないサイトよりもサードパーティCookieが大幅に多くなっています。
- 最も人気のあるCookieは広告、トラッキング、アナリティクスのユースケースと関連付けられています。
- Cookieは中央値の平均寿命が12ヶ月と長寿命になる傾向があります。 一時的な

509. *https://developer.chrome.com/blog/cookie-max-age-expires*
510. *https://almanac.httparchive.org/ja/2024/cookies#まとめ*

セッションCookieはファーストパーティの19%、サードパーティの7%のみを占めます。

- Cookieの機能に対するその他の制限はほとんど使用されないか、まったく使用されません。サードパーティCookieの10%がパーティション化されている（昨年の6%からわずかに増加）一方、サードパーティCookieの100%がクロスサイトリクエストで送信できる`SameSite=None`を持っています。さらに、Cookieプレフィックスの採用はほぼ存在しません。

最後に、プライバシーの懸念から複数のウェブブラウザがサードパーティCookieを廃止または制限[511]している一方、GoogleはChromeでのサードパーティのサポートを継続する[512]ことを決定しました。Googleはまた、当初*「ユーザーを尊重し、デフォルトでプライバシーを保護するウェブエコシステムの繁栄を生み出す」*ために設計されたPrivacy Sandboxイニシアチブのほとんどの技術を段階的に廃止しています。その結果、トラッカーがサードパーティCookieを使用するか、オンラインでユーザーを追跡するために他の技術（ファーストパーティ同期、フィンガープリンティングなど）を開発するかにかかわらず、Cookieはウェブにとってプライバシーとセキュリティリスクをもたらし続ける基本的なコンポーネントであり続けます。

著者

Yohan Beugin

yohhaan https://yohan.beugin.org

Yohan BeuginはWisconsin大学マディソン校のコンピュータサイエンス学科の博士課程の学生で、セキュリティとプライバシー研究グループのメンバーであり、Patrick McDaniel教授の指導を受けています。より安全で、プライバシーを保護し、信頼できるシステムの構築に関心を持っています。現在の研究はオンライン広告のトラッキングとプライバシー、そしてオープンソースソフトウェアのセキュリティに焦点を当てています。

511. *https://developer.mozilla.org/docs/Web/Privacy/Guides/Third-party_cookies#how_do_browsers_handle_third-party_cookies*
512. *https://privacysandbox.com/news/update-on-plans-for-privacy-sandbox-technologies/*

付属資料 A
方法論

概要

Web Almanacは、HTTP Archive[513]によって組織されたプロジェクトです。HTTP Archiveは、2010年にSteve Soudersによってウェブがどのように構築されているかを追跡する使命で始まりました。月に数百万のウェブページの構成を評価し、そのテラバイトのメタデータをBigQuery[514]で分析用に提供しています。

The Web Almanacの使命は、Webの現状に関する公共知識の年間リポジトリになることです。我々の目標は、HTTP ArchiveのデータウェアハウスをWebコミュニティにとってさらに

513. *https://httparchive.org*
514. *https://httparchive.org/faq#how-do-i-use-bigquery-to-write-custom-queries-over-the-data*

アクセスしやすくすることであり、専門家がコンテキストに沿った洞察を提供することでこれを実現します。

2025年版のWeb Almanacは、コンテンツ、エクスペリエンス、パブリッシング、ディストリビューションの4つの部分に分かれています。それぞれの部分では、複数の章がさまざまな角度からその全体的なテーマを探ります。たとえば、第2部では、パフォーマンス、セキュリティ、アクセシビリティの章などでユーザーエクスペリエンスを異なる視点から探求しています。

データセットについて

HTTP Archiveのデータセットは、毎月新しいデータで継続的に更新されています。2025年版のWeb Almanacでは、とくに章で記載がない限り、すべての指標は2025年7月のクロールから取得されました。これらの結果は、BigQuery上の `` `httparchive.crawl.*` `` テーブルの `date = '2025-07-01'` で公開クエリが可能[515]です。

Web Almanacで提示されているすべての指標は、BigQuery上のデータセットを使用して公開再現可能です。すべての章で使用されたクエリは、GitHubリポジトリ[516]で閲覧できます。

これらのクエリの一部は非常に大きく、自分で実行する場合、高額[517]になる可能性がありますのでご注意ください。費用を抑える方法については、クエリコストの最小化[518]ガイドを参照してください。

たとえば、アクセシブルなカラーコントラストを持つページ数を理解するには、color_contrast.sql[519]を参照してください：

```
SELECT
  client,
  is_root_page,
  COUNTIF(color_contrast_score IS NOT NULL) AS
total_applicable,
  COUNTIF(CAST(color_contrast_score AS NUMERIC) = 1) AS
total_good_contrast,
```

515. *https://har.fyi/guides/getting-started/*
516. *https://github.com/HTTPArchive/almanac.httparchive.org/tree/main/sql/2025*
517. ***https://cloud.google.com/bigquery/pricing***
518. ***https://har.fyi/guides/minimizing-costs/***
519. *https://github.com/HTTPArchive/almanac.httparchive.org/blob/main/sql/2025/accessibility/color_contrast.sql*

```
    COUNTIF(CAST(color_contrast_score AS NUMERIC) = 1) /
  COUNTIF(color_contrast_score IS NOT NULL) AS
  perc_good_contrast
  FROM (
    SELECT
      client,
      is_root_page,
      date,
      JSON_VALUE(lighthouse.audits.`color-contrast`.score) AS
  color_contrast_score
    FROM
      `httparchive.crawl.pages`
    WHERE
      date = '2025-07-01'
  )
  GROUP BY
    client,
    is_root_page,
    date
  ORDER BY
    client,
    is_root_page;
```

各指標の結果は、たとえばアクセシビリティの結果[520]など、章ごとのスプレッドシートで公開されています。各章の末尾には、生データとクエリへのリンクが記載されています。また、指標ごとの結果とクエリは各フィギュアから直接リンクされています。

ウェブサイト

データセットには16,213,084のウェブサイトが含まれています。そのうち、15,426,398はモバイルウェブサイトで、12,155,374はデスクトップウェブサイトです。ほとんどのウェブサイトは、モバイルおよびデスクトップの両方のサブセットに含まれています。

HTTP Archiveは、Chrome UX ReportからウェブサイトのURLを取得しています。Chrome UX Reportは、Googleによる公開データセットで、Chromeユーザーが積極的に訪問してい

520. https://docs.google.com/spreadsheets/d/13sY_wmYODArxo-hH5cSuRAbvtLdGI3x5Xc9qUyqP8as

る何百万ものウェブサイトにおけるユーザーエクスペリエンスを集約しています。これにより、最新の実際のWeb使用状況を反映したウェブサイトのリストを入手できます。Chrome UX Reportのデータセットにはフォームファクターの次元が含まれており、これを使用してデスクトップまたはモバイルユーザーがアクセスしたすべてのウェブサイトを取得しています。

Web Almanacで使用された2025年7月のHTTP Archiveクロールは、もっとも最近利用可能なChrome UX Reportリリースをウェブサイトのリストとして使用しました。202507のデータセットは2025年7月11日にリリースされ、6月にChromeユーザーが訪問したウェブサイトをキャプチャしています。

リソースの制約により、HTTP Archiveは以前、Chrome UX Reportの各ウェブサイトから2ページしかテストできませんでした。ホームページは必ずしもウェブサイト全体を代表しているわけではないため、結果にバイアスがかかる可能性があることに注意してください。近年、セカンダリページ[521]を導入し、多くの章でこの新しいデータを使用しています。しかし、一部の章ではホームページのみを使用しました。

HTTP Archiveは、ラボテストツールとも見なされており、データセンターからウェブサイトをテストし、実際のユーザーエクスペリエンスからのデータを収集していません。すべてのページは、キャッシュが空の状態で、ログアウトした状態でテストされており、実際のユーザーがアクセスする方法を反映していない場合があります。

メトリクス

HTTP Archiveは、Webがどのように構築されているかについて何千ものメトリクスを収集しています。これには、ページごとのバイト数、ページがHTTPSで読み込まれたかどうか、個々のリクエストおよびレスポンスヘッダーなどの基本的なメトリクスが含まれます。これらのメトリクスの大部分は、各ウェブサイトのテストランナーとして機能するWebPageTestによって提供されます。

他のテストツールは、ページに関するより高度なメトリクスを提供するために使用されます。たとえば、Lighthouseは、アクセシビリティやSEOなどの分野でページの品質を分析するための監査を実行するために使用されます。これらのツールの詳細については、下のツールセクションで説明しています。

ラボデータセットの固有の制約を回避するために、Web Almanacは、とくにウェブパフォーマンスの分野でのユーザーエクスペリエンスに関するメトリクスのためにChrome UX Reportも使用しています。

一部のメトリクスは完全に把握できません。たとえば、ウェブサイトを構築するために使用

521. *https://discuss.httparchive.org/t/improving-the-http-archive-pipeline-and-dataset-by-10x/2372*

されたツールを検出する能力は必ずしも持っていません。create-react-appを使用してウェブサイトが構築されている場合、そのサイトがReactフレームワークを使用していることはわかりますが、特定のビルドツールが使用されているかどうかは必ずしもわかりません。これらのツールがウェブサイトのコードに検出可能な指紋を残さない限り、それらの使用を測定することはできません。

他のメトリクスは測定が不可能ではないにしても、挑戦的であるか、信頼性が低いことがあります。たとえば、ウェブデザインの側面は本質的に視覚的であり、ページに煩わしいモーダルダイアログがあるかどうかなど、定量化が困難です。

ツール

Web Almanacは、次のオープンソースツールの助けを借りて実現されています。

WebPageTest

WebPageTest[522]は、著名なウェブパフォーマンステストツールであり、HTTP Archiveの基盤です。私たちは、プライベートインスタンス[523]のWebPageTestを使用しており、各ウェブページをテストするためのプライベートテストエージェント（実際のブラウザ）を使用しています。デスクトップおよびモバイルウェブサイトは、異なる構成でテストされています：

522. *https://www.webpagetest.org/*
523. *https://docs.webpagetest.org/private-instances/*

設定	デスクトップ	モバイル
デバイス	Linux VM	エミュレートされたMoto G4
ユーザーエージェント	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36 PTST/250604.160032	Mozilla/5.0 (Linux; Android 8.1.0; Moto G (4)) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Mobile Safari/537.36 PTST/250604.160032
ロケーション	Google Cloud Locations, USA	Google Cloud Locations, USA
接続	ケーブル (5/1 Mbps 25ms RTT)	4G (9 Mbps 170ms RTT)
CPUの速度低下	*N/A*	8x
ビューポート	1920 x 1080px	512 x 360px
DPR	1x	3x

デスクトップウェブサイトは、Linux VM上のデスクトップChrome環境内で実行されます。ネットワーク速度はケーブル接続に相当します。

モバイルウェブサイトは、エミュレートされたMoto G4デバイス上のモバイルChrome環境内で実行され、ネットワーク速度は4G接続に相当します。

テストエージェントは、米国に拠点を置くさまざまなGoogle Cloud Platformロケーション[524]から実行されます。

HTTP ArchiveのWebPageTestのプライベートインスタンスは、最新のパブリックバージョンと同期されており、カスタムメトリクス[525]が追加されています。これらは、各ウェブサイトでテストの終了時に評価されるJavaScriptのスニペットです。

各テストの結果は、ウェブページに関するメタデータを含むJSON形式のアーカイブファイルであるHARファイル[526]として利用可能であり、BigQueryに保存されます。

524. *https://cloud.google.com/compute/docs/regions-zones/#locations*
525. *https://github.com/HTTPArchive/custom-metrics*
526. *https://en.wikipedia.org/wiki/HAR_(file_format)*

Lighthouse

Lighthouse[527]は、Googleによって開発された自動ウェブサイト品質保証ツールです。ウェブページを監査し、最適化されていない画像やアクセシビリティのないコンテンツなど、ユーザーエクスペリエンスに反するパターンが含まれていないことを確認します。

HTTP Archiveは、すべてのページで最新バージョンのLighthouseを実行しています。2025年7月のクロール時点で、HTTP Archiveは12.8.0[528]バージョンのLighthouseを使用しました。

Lighthouseは、WebPageTest内から独自のテストとして実行されますが、独自の設定プロファイルを持っています：

設定	デスクトップ	モバイル
CPUの速度低下	*N/A*	1x/4x
ダウンロードのスループット	10.0 Mbps	1.6 Mbps
アップロードのスループット	10.0 Mbps	0.75 Mbps
RTT	40 ms	150 ms

LighthouseおよびHTTP Archiveで利用可能な監査に関する詳細は、Lighthouse開発者ドキュメント[529]を参照してください。

Wappalyzer

Wappalyzer[530]は、ウェブページで使用されている技術を検出するツールです。JavaScriptフレームワーク、CMSプラットフォーム、さらには暗号通貨マイナーまで、108のカテゴリ[531]にわたる技術がテストされています。サポートされている技術は3,984を超えています。

HTTP Archiveは、最後のオープンソース版Wappalyzer（v6.10.65）のフォークを実行しており、その後いくつかの追加検出機能が加えられています。

Wappalyzerは、WordPress、Bootstrap、jQueryなどの開発者ツールの人気を分析する多くの章で利用されています。たとえば、CMS章では、Wappalyzerが検出したCMS[532]カテゴリの技術に大きく依存しています。

Wappalyzerを含むすべての検出ツールには制限があります。その結果の正確性は、検出メカ

527. *https://developer.chrome.com/docs/lighthouse/overview*
528. *https://github.com/GoogleChrome/lighthouse/releases/tag/12.8.0*
529. *https://developer.chrome.com/docs/lighthouse/overview*
530. *https://github.com/HTTPArchive/wappalyzer*
531. *https://github.com/HTTPArchive/wappalyzer/blob/main/src/categories.json*
532. *https://www.wappalyzer.com/categories/cms*

ニズムの精度に依存します。Web Almanacは、Wappalyzerが使用されるすべての章において、特定の理由で分析が正確でない可能性がある場合、その旨を注記します。

Chrome UX Report

Chrome UX Report[533]は、実際のChromeユーザーのエクスペリエンスに関する公開データセットです。エクスペリエンスは、ウェブサイトのオリジンごとにグループ化されます。たとえば、`https://www.example.com`です。このデータセットには、ペイント、ロード、インタラクション、レイアウト安定性などのUX指標の分布が含まれています。月ごとにグループ化されるだけでなく、国レベルの地理、フォームファクター（デスクトップ、電話、タブレット）、有効な接続タイプ（4G、3Gなど）などの次元で分割することもできます。

Chrome UX Reportのデータセットには、相対的なウェブサイトランキングデータ[534]も含まれています。これらは *ランクマグニチュード* と呼ばれ、1位や116位のような細かな順位ではなく、上位1k、上位10k、最大で上位10Mまでのランクバケットにウェブサイトがグループ化されます。各ウェブサイトは、すべてのページを合わせた対象となる[535]ページビューの数に基づいてランク付けされます。今年のWeb Almanacでは、この新しいデータを広範に使用し、サイトの人気によってWebの構築方法にどのような違いがあるかを探る手段としています。

Web Almanacのメトリクスで、Chrome UX Reportからの実際のユーザーエクスペリエンスデータを参照しているものには、2025年7月のデータセット（202507）が使用されています。

データセットの詳細については、Using the Chrome UX Report on BigQuery[536]ガイドをdeveloper.chrome.com[537]でご覧ください。

Blink Features

Blink Features[538]は、特定のWebプラットフォーム機能が使用されていると検出された際にChromeがフラグを立てる指標です。

私たちは、Blink Featuresを使用して機能の採用状況を別の視点から捉えています。このデータは、ページに実装されている機能と実際に使用されている機能を区別するのにとくに有用です。

Blink Featuresは、WebPageTestの通常のテストの一環として報告されています。

533. *https://developer.chrome.com/docs/crux*
534. *https://developer.chrome.com/blog/crux-rank-magnitude*
535. *https://developer.chrome.com/docs/crux/methodology#eligibility*
536. *https://developer.chrome.com/docs/crux/guides/bigquery*
537. *https://developer.chrome.com*
538. *https://chromium.googlesource.com/chromium/src/+/HEAD/docs/use_counter_wiki.md*

Third Party Web

Third Party Web[539]は、HTTP ArchiveおよびLighthouseデータを使用して、Web上のサードパーティリソースの影響を特定し分析するPatrick Hulce（2019年のサードパーティ章の著者）による研究プロジェクトです。

ドメインは、少なくとも50の異なるページに表示される場合、サードパーティプロバイダーと見なされます。このプロジェクトでは、プロバイダーを広告、分析、ソーシャルなどのカテゴリごとにグループ化しています。

Web Almanacのいくつかの章では、このデータセットのドメインとカテゴリを使用して、サードパーティの影響を理解しています。

Rework CSS

Rework CSS[540]は、JavaScriptベースのCSSパーサーです。スタイルシート全体を取り込み、各スタイルルール、セレクター、ディレクティブ、値を区別するJSON形式のオブジェクトを生成します。BigQuery上のHTTP Archiveデータセットとの統合方法については、このスレッド[541]を参照してください。

Parsel

Parsel[542]は、CSSセレクターパーサーおよび特異性計算機で、元々は2020年のCSS章のリーダーであるLea Verouによって書かれ、別のライブラリとしてオープンソース化されました。これは、セレクターや特異性に関連するすべてのCSS指標で広く使用されています。

分析プロセス

Web Almanacは、ウェブコミュニティの100人以上の寄稿者との調整を伴い、計画および実行に約1年かかりました。このセクションでは、Web Almanacで選ばれた章がどのような理由で選ばれ、その指標がどのようにクエリされ、どのように解釈されたかを説明します。

計画

2025年のWeb Almanacは、2025年4月に寄稿者募集の呼びかけ[543]でスタートしました。プロ

539. https://www.thirdpartyweb.today/
540. https://github.com/reworkcss/css
541. https://discuss.httparchive.org/t/analyzing-stylesheets-with-a-js-based-parser/1683
542. https://projects.verou.me/parsel/
543. https://github.com/HTTPArchive/almanac.httparchive.org/discussions/4062

ジェクトは前年と同じ章でスタートし、コミュニティから提案された追加のトピックの中から、今年の新しい1章となりました：生成AI。

私たちは創刊年の方法論で述べた通り：

> *Web Almanacの将来の版の1つの明確な目標は、著者やピアレビュアーとして代表されていない多様な声をさらに奨励することです。*

そのために、今年も以下の方針のもと著者選定プロセス[544]を続けています：

- 新しい著者には、異なる視点をもたらすため、積極的に参加を奨励しました。
- プロジェクトリーダーはすべての著者の推薦をレビューし、新しい視点をもたらし、コミュニティ内の代表されていないグループの声を増幅する著者を選出する努力をしました。

私たちは、Web Almanacがさらに多様で包括的なプロジェクトとなるよう、このプロセスを将来的に反復し、すべての背景を持つ寄稿者からの意見を取り入れることを望んでいます。

分析

2025年5月と6月に、データアナリストは著者およびピアレビュアーと協力して、各章でクエリを実行する必要がある指標のリストを作成しました。場合によっては、分析能力のギャップを埋めるためにカスタムメトリクス[545]が作成されました。

2025年7月を通じて、HTTP Archiveのデータパイプラインは数百万のウェブサイトをクロールし、Web Almanacで使用されるメタデータを収集しました。これらの結果は後処理され、BigQuery[546]に保存されました。

第6回目となる今年、以前のアナリストによって書かれたクエリを更新して再利用することができましたが、HTTP Archiveが今年移行した新しいより効率的なデータセット[547]に合わせてクエリを書き直す必要がありました。また、ゼロから書かれる必要がある多くの新しい指標もありました。GitHub上のオープンソースクエリリポジトリ[548]で、年度および章ごとのすべてのクエリを閲覧できます。

解釈

著者はアナリストと協力して結果を正しく解釈し、適切な結論を導き出しました。著者がそ

544. *https://github.com/HTTPArchive/almanac.httparchive.org/discussions/2165*
545. *https://github.com/HTTPArchive/custom-metrics*
546. *https://console.cloud.google.com/bigquery?p=httparchive&d=almanac&page=dataset*
547. *https://har.fyi/guides/migrating-to-crawl-dataset/*
548. *https://github.com/HTTPArchive/almanac.httparchive.org/tree/main/sql/2025*

れぞれの章を書く際には、これらの統計を利用してウェブの状態を枠組みとして支えました。ピアレビュアーは著者と協力して分析の技術的正確さを保証しました。

結果を読者により容易に理解してもらうために、ウェブ開発者およびアナリストは章に埋め込むデータビジュアリゼーションを作成しました。いくつかのビジュアライゼーションは、ポイントをより明確にするために簡略化されています。たとえば、完全な分布を示すのではなく、いくつかのパーセンタイルのみが表示されます。とくに記載がない限り、すべての分布はパーセンタイルを使用して要約され、とくに中央値（50パーセンタイル）が使用され、平均は使用されません。

最後に、編集者が章を見直して単純な文法的な誤りを修正し、読書体験全体の一貫性を保証しました。

将来に向けて

2025年版のWeb Almanacは、ウェブコミュニティの内省と前向きな変化への取り組みとして（2023年は一時休止しましたが）ほぼ年次の伝統における第6回目です。この地点に到達するまで、多くの献身的な寄稿者のおかげで壮大な努力がなされました。私たちは、将来の版をさらに効率的にするために、可能な限りこの作業を活用することを望んでいます。

付属資料 B
貢献者






Web Almanacは、ウェブ・コミュニティの努力によって実現しています。2025 Web Almanac々は、企画、調査、執筆、制作の段階で75人が数え切れないほどの時間をボランティアで費やしてきました。

Aaron Gustafson
@AaronGustafson
aarongustafson
https://www.aaron-gustafson.com
レビュワー

Aaron T. Grogg
aarontgrogg.com
aarontgrogg
aarontgrogg
https://aarontgrogg.com/
著作者

Aidan Tierney
aidantierney.bsky.social
AidanA11y
aidantierney
レビュワー

Alba Silvente Fuentes
@dawntraoz
Dawntraoz
https://www.dawntraoz.com/
著作者

Alex Moening
AlexMoening
分析者 と 著作者

Alon Kochba
@alonkochba
alonkochba
alonkochba
分析者

Amaka Chulwuma
Amaka-maxi
amaka-chukwuma-jubilee
著作者

Amandeep Singh
@amanvirdi
amandeepsinghvirdi
amandeepsinghvirdi
https://byaman.com/
分析者 と 著作者

Anirudh Duggal
anirudhduggal
anirudh-duggal
レビュワー

Augustin Delport
分析者

Barry Pollard
@tunetheweb
https://webperf.social/@tunetheweb
tunetheweb.com
tunetheweb
tunetheweb
https://www.tunetheweb.com
分析者、 組織委員会、 開発者、 編集者 と レビュワー

Bobo Chen

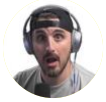
bobo52310
https://linktr.ee/bobo.tips
翻訳者

Bogdan Lazar

bogdanlazar.com
tricinel
tricinel
https://bogdanlazar.com
著作者 と 編集者

Bram Stein

https://typo.social/@bram
bramstein
bramstein
http://www.bramstein.com/
レビュワー



Brian Clark

@_clarkio
clarkio
https://www.clarkio.com
レビュワー

Brian Smith

https://mastodon.social/@bsmth
bsmth.de
bsmth
https://bsmth.de
編集者

Burak Güneli

@burakGuneli
burakguneli
組織委員会

Caroline Scholles

carolinescholles
編集者 と レビュワー

Charles Berret

charlesberret
https://charlesberret.net/
分析者 と 著作者

Chris Böttger

ChrisBeeti
分析者 と 組織委員会

Chris Green

@chrisgreenseo
chris-green.net
chr156r33n
chrisgreenseo
https://www.torquepartnership.com/
分析者 と 著作者

Chris Lilley

@svgeesus
svgeesus
https://svgees.us
編集者

Christian Liebel

@christianliebel
https://mastodon.cloud/@christianliebel
christianliebel
christianliebel
https://christianliebel.com
分析者 と 著作者

Daan Vansteenhuyse

vsdaan
daan-vansteenhuyse-151297371
https://vsdaan.be
分析者

Dan Knauss

dknauss
https://newlocalmedia.com
レビュワー

Dave Smart

tamethebots.com
dwsmart
davewsmart
https://tamethebots.com
分析者 と レビュワー

Diego Gonzalez

@diekus
diekus
https://diek.us
著作者

Estela Franco

@guaca
https://toot.cafe/@guaca
guaca.bsky.social
guaca
estelafranco
https://estelafranco.com/
分析者

Gertjan Franken

@GJFR_
GJFR
gertjan-franken
https://gjfr.dev
著作者 と レビュワー

Hidde de Vries

https://front-end.social/@hdv
hidde
https://hidde.blog/
レビュワー

Himanshu Jariyal

himanshujariyal
himanshujariyal
https://medium.com/@him_jar
著作者

Humaira

hfhashmi
著作者

Ivan Ukhov

IvanUkhov
分析者

Jamie Indigo

not-a-robot.com
fellowhuman1101
jamie-indigo
https://not-a-robot.com
著作者 と レビュワー

Jannis Rautenstrauch

@jannis_r
JannisBush
jannis-rautenstrauch
https://jannisbush.github.io/
レビュワー

Joe Viggiano

joeviggiano
分析者 と 著作者

Jonathan Pagel

jcmpagel
jonathan-pagel
https://jonathanpagel.com
分析者 と 著作者

José Solé

@jmsoleb
jmsole
https://www.jmsole.cl/
レビュワー

Kai Hollberg

https://mastodon.social/@Schweinepriester
Schweinepriester
レビュワー

Martina Kraus

@MartinaKraus11
martinakraus
martina-kraus-398493108
レビュワー

Max Ostapenko

max-ostapenko
max-ostapenko
https://maxostapenko.com
分析者 と 編集者

Maxim Salnikov

@webmaxru
webmaxru
https://medium.com/@webmaxru
レビュワー

Michael Lewittes

@MichaelLewittes
https://seocommunity.social/@MichaelLewittes
michaellewittes.bsky.social
MichaelLewittes
michael-lewittes-a22b831
https://www.ranktify.com/team
編集者

Michael Solati

@MichaelSolati
MichaelSolati
https://michaelsolati.com
著作者

Mika Steinleitner

mika943
翻訳者

Mikael Araújo

@miknaraujo
mikaelaraujo.bsky.social
mikaelaraujo
mikael-araujo
https://www.mikaelaraujo.com
組織委員会

Mike Gifford

https://mastodon.social/@mgifford
mgifford.bsky.social
mgifford
mgifford
https://accessibility.civicactions.com/
分析者 と 著作者

Montserrat Cano

montsecano.bsky.social
montsec
montsecano-senior-digital-marketer
https://montserrat-cano.com/
編集者

Muhammad Abu Bakar Aziz

abubakaraziz
aziz313f
著作者

Muhammad Jazlan

jazlan01
分析者 と 著作者

Nick McCord

nick-mccord
nicholasmccord
分析者 と 著作者

Nimesh Vadgama - VN

nimeshgit
ops-ml-architect
https://ops-ml-architect.blogspot.com/
分析者 と 著作者

Nurullah Demir

nrllh
nurullahdemir
https://ndemir.com
著作者、 組織委員会、 リーダー と レビュワー

Onur Güler

onurglr
分析者

Prathamesh Rasam

25prathamesh
prathamesh-rasam
著作者

Richard Barret

barkseo.bsky.social
rickb110
richardbarrettseo
著作者

Rumaisa Habib

RumaisaHabib
rumaisahabib
https://rumaisahabib.com/
著作者

Sakae Kotaro

@beltway7
ksakae1216
https://ksakae1216.com/
翻訳者

Samuel Fatola

samuelfatola
編集者

Saptak Sengupta

@Saptak013
SaptakS
https://saptaks.website
組織委員会 と 開発者

Scott Davis

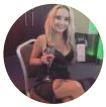
@scottdavis99
scottdavis99
http://thirstyhead.com
レビュワー

Selman Koral

krlslman
組織委員会、 開発者 と 翻訳者

Sharon McClintic

hello-sharon
sharonmcclintic
https://www.lumar.io/
編集者



Sophie Brannon

@SophieBrannon
SophieBrannon
著作者

Sreemoyee Bhattacharya

sreemoyee-bhattacharya-17a4a61a6
編集者

Stoyan Stefanov

@stoyanstefanov
https://indieweb.social/@stoyan
stoyan.me
stoyan
stoyanstefanov
https://www.phpied.com
レビュワー

Tanner Hodges

tannerhodges.bsky.social
tannerhodges
https://tannerhodges.com/
分析者 と レビュワー

Thomas Steiner

@tomayac
tomayac
https://blog.tomayac.com/
分析者 と レビュワー

Tobias Urban

turban1988
https://internet-sicherheit.de/ueber-uns/team/alle-mitarbeiter/urban-tobias-2/
組織委員会

Umar Iqbal

@umaarr6
UmarIqbal
https://www.umariqbal.com
レビュワー

Vahe Arabian

VaheSODP
ahearabian
https://www.stateofdigitalpublishing.com/
著作者

Vik Vanderlinden

@vikvanderlinden
vikvanderlinden
vikvanderlinden
https://vikvanderlinden.be/
著作者

Vinod Tiwari

@@securient
securient
https://blog.securient.io
著作者 と レビュワー

Yash Vekaria

@vekariayash
Yash-Vekaria
https://www.yashvekaria.com/
著作者 と 組織委員会

Yohan Beugin

yohhaan
https://yohan.beugin.org
著作者