プレスリリース

2016 年 3 月 28 日

報道関係者各位

慶應義塾大学 SFC 研究所

HTML5 サイネージ端末の性能評価指標を策定

～性能要件・性能テスト仕様・コンテンツ開発ガイドラインを発行～

慶應義塾大学 SFC 研究所 先端ウェブアーキテクチャ・ラボ(代表:村井純 環境情報学部長・教授、執行代表:中村修 環境情報学部教授)では、このたび、HTML5 ベースのデジタルサイネージ端末の表示性能に関する要求条件、性能テスト仕様、およびコンテンツ開発ガイドラインを、本日発行いたします。

これらのドキュメントは、ウェブプラットフォームの発展と産業市場拡大を目的として性能テストによるウェブ技術の底上げ・品質向上を目指し、当ラボも主査として参画している「Web プラットフォーム性能ベンチマーク検討会」(※)が、(一社)デジタルサイネージコンソーシアム(理事長:中村伊知哉 慶應義塾大学大学院メディアデザイン研究科教授)の協力を得て、作成したものです。ドキュメントと合わせ、性能テスト仕様に基づいたテスト用 HTML5 コンテンツも作成しました。

今後も当ラボでは、オープンウェブプラットフォームの確立に向け、産業界での HTML5 の活用がより一層拡大するための活動を行ってまいります。

【概要】

セットトップボックスなどをはじめとする組込系のデバイスでは、PC と異なり、CPU やメモリの制約から、ウェブ表示の際の性能が重要な検討項目になります。この性能に対する要求条件は、デバイスの利用用途により異なるため、分野ごとに定義する必要があります。

そこでこのたび、今後普及が予想される HTML5 サイネージ(Web-based Signage)を対象として、以下を作成しました。これらは、主として端末の HTML5 表示性能評価に関するもの(1)(2)(4)と、その性能を活かすための HTML5 コンテンツ作成ノウハウに関するもの(3)に分けてあります。

(1)端末性能要件(資料 1 http://www.kri.sfc.keio.ac.jp/ja/press_file/20160328_awalab.pdf)

- HTML5 サイネージ端末の性能への要求条件定義。対象は、解像度・ストレージ・描画速度など7項目の表示関連の性能である。各項目ごとに性能が定量的に 2～3 クラスに分けられている。

(2)端末性能テスト仕様(資料 2 http://www.kri.sfc.keio.ac.jp/ja/press_file/20160328_awalab.pdf)

- 要件に基づき HTML5 サイネージ端末を性能テストするための仕様書

(3)コンテンツ開発ガイドライン(資料 3 http://www.kri.sfc.keio.ac.jp/ja/press_file/20160328_awalab.pdf)

- 端末の性能を引き出すための HTML5 コンテンツ開発ガイドライン

(4)性能テスト用コンテンツ

- 端末性能テスト仕様に基づいた HTML5 テストコンテンツ

これらは、次の図に示すように、HTML5 サイネージに関係する様々な事業者の方々に活用いただけます。例えば、端末メーカーは、性能の「見える化」により、自社の HTML5 サイネージ端末のラインナップ化に活用可能です。またロケーションオーナーや広告主は、コンテンツに適した端末の選定に役立てることができます。

当ラボでは、HTML5 サイネージの普及のため、関係者に広くこれらを活用いただけるよう、引き続き活動してまいります。

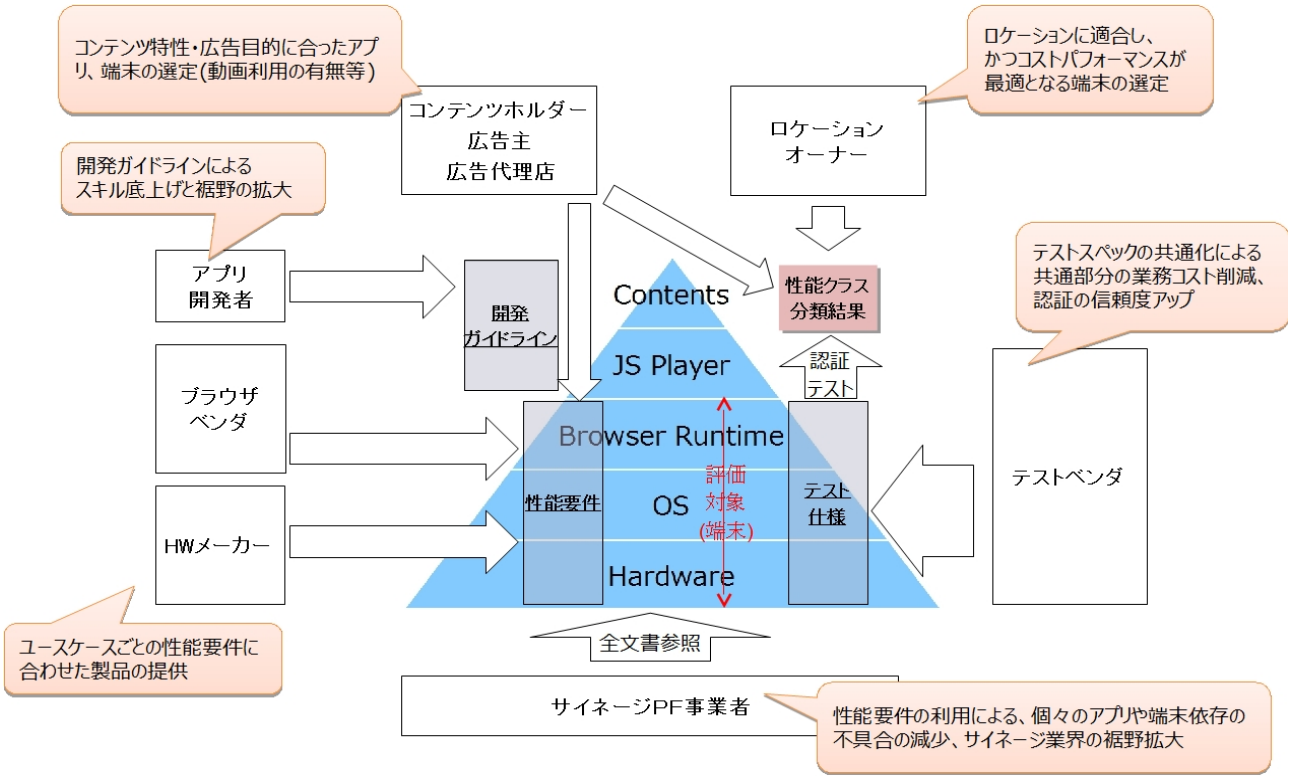


図 1: 各ドキュメントの活用方法

【(※)Web プラットフォーム性能ベンチマーク検討会について】
ウェブプラットフォームの発展と産業市場拡大を目的として、2014 年 10 月より、性能テスト実施による技術の底上げ・品質向上の方策を検討。メンバーは次のとおり(50 音順)。
(株)ACCESS、キャッツ(株)、慶應義塾大学 SFC 研究所 先端ウェブアーキテクチャ・ラボ(主査)、(株)シーイーシー、(株)東芝、(株)ニューフォリア

【ご参考】
添付資料 1: Web-based Signage 端末性能要件第 1 版
添付資料 2: Web-based Signage 端末性能テスト仕様第 1 版
添付資料 3: Web-based Signage コンテンツ開発ガイドライン第 1 版

本発表資料のお問い合わせ先
慶應義塾大学 SFC 研究所
先端ウェブアーキテクチャ・ラボ HTML5 サイネージ端末性能評価担当
TEL 03-3516-2504 Email: lab-contacts@awa.sfc.keio.ac.jp

慶應義塾大学湘南藤沢事務室学術研究支援担当
TEL 0466-49-3436 Email: kri-pr@adst.keio.ac.jp

資料 1

Web-based Signage 端末
性能要件

第 1 版

2016 年 3 月

作成： Web プラットフォーム性能ベンチマーク検討会

発行： 慶應義塾大学 SFC 研究所 先端ウェブアーキテクチャ・ラボ

目次

1 性能要求条件の概要

本文書は、Web-based Signage(ウェブ/HTML5 技術に基づくデジタルサイネージ)普及を目的に、Web-based Signage に求められるプレーヤー端末の性能に関して、ウェブ/HTML5 技術関連の要求条件を整理したものである。本文書では、３章から１０章に各要求条件が定義されている。各要件ごとの評価テスト手順については、「Web-based Signage テストスペック」文書に記されている。

1.1 性能評価の範囲

Web-based Signage は、コンテンツを管理する CMS（コンテンツ登録やプレイリストなどを管理）、コンテンツを配信する CDN（配信クラウドとネットワーク）、そして、コンテンツを再生する端末から構成される。本資料は、これら Web-based Signage システム全体のうち、端末を評価の対象にする。

端末は、以下の要素で構成される。

ハードウェア

物理的に端末を構成する構成要素を指す。具体的には、パネル、SoC（CPU, GPU, メモリなど）、入出力インタフェース（HDMI, USB, 電源など）で構成される。

オペレーティングシステム

Web-based Signage として端末ハードウェアを利用するためにインストールされる基本ソフトを指す。具体的には、Windows, Android, iOS, Linux, FreeBSD, Firefox OS などを指す。

ブラウザーランタイム

HTML, CSS, JavaScript などの Web 標準技術を動作させることができるアプリケーション基盤を指す。具体的にはウェブブラウザーを指すが、必ずしも、一般的なウェブブラウザーである必要はない。例えば、Android の WebView など、ランタイムとして用意された環境も含む。

JS プレーヤー

プレイリストに基づいてコンテンツを再生するためのアプリケーションを指す。Web-based Signage では、ブラウザーランタイム上で動作するウェブアプリケーションとして作られる。具体的には、HTML, CSS, JavaScript を駆使して開発される。

コンテンツ

コンテンツは、JS プレーヤー上で表示または実行され、サイネージ端末の前にいるユーザーに見えるものである。コンテンツは、単体画像、動画、または、ウェブアプリケーションとして制作される。中には、これらコンテンツを表示しつつ、音声を再生するものも存在する。

本資料の性能評価の対象である端末とは、上記構成要素のうち、ハードウェア、オペレーティングシステム、ブラウザーランタイムまでを意味する。

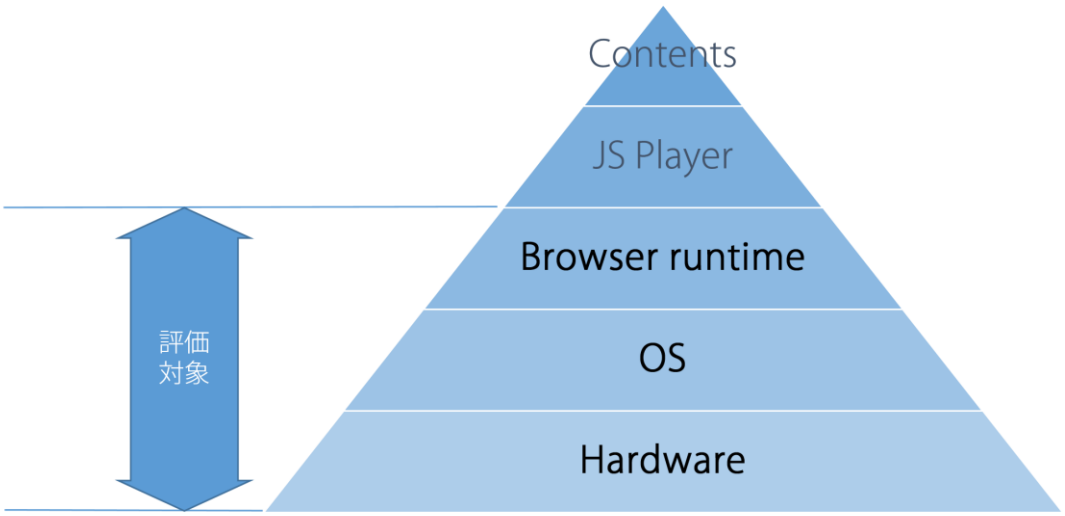
また、本性能評価は、これら構成要素を個別に評価するのではなく、これらを組み合わせた状態、つまり、実際の端末商品として総合的に評価することとする。

なお、ウェブ/HTML5 技術を使ったサイネージ端末のうち、ネットワーク接続を有しない、または、ネットワーク接続を有するもののネットワークを使わずにコンテンツを再生するスタンドアローン型のサイネージ端末（USB ストレージに保存したコンテンツを再生するタイプの端末）は、評価の対象外とする。

本性能要件文書の記載対象は、ウェブ/HTML5 の表示関連の性能のみに絞っている。例えば長期安定性（ハングアップしないこと）やタスクスケジューラ関連など、サイネージ端末に対する一般的な要件については対象としない。また、外部デバイスとの連携機能(デバイス API など)については、現状のブラウザでの実装状況をかんがみ、本版では対象としていない。

1.2 評価の方針

Web-based Signage の用途は多岐にわたる。必ずしも、端末が本資料の性能要求条件すべてを満たす必要はない。そのため、本資料では、端末を単一的なレベル評価ではなく、性能レベルを端末の用途に応じた適合用途として分類し、それぞれの適合要件を定義することとする。



2 適合要件の分類と性能評価

端末の本評価要件では、まず評価分類を定義し、その評価分類ごとに性能を軸とした適合分類に端末を分類する。分類の概要は下表のとおり。

分類表

評価分類	適合分類	説明
解像度	小密度型	離れた場所から見ることを前提にした旧来型のサイネージを指す。小さな文字を表示するコンテンツには向かない。
	中密度型	至近距離から見られることを前提に、小さな文字（映画の字幕より小さい文字）でも明瞭に読めるレベルのサイネージを指す。また、パネルのサイズが小さければ、ブランディングを重視したコンテンツにも使われる。
	高密度型	ブランディングやリアリティを重視したコンテンツを表示できるレベルのサイネージを指す。
ストレージ	キャッシュ型	コンテンツとなる画像ファイルや動画ファイルを保存する機能を有しない端末を指す。
	蓄積型	コンテンツとなる画像ファイルや動画ファイルを保存する機能を有する端末を指すが、保存サイズが小さいものを指す。
	大容量蓄積型	コンテンツとなる画像ファイルや動画ファイルを保存する機能を有する端末を指すが、保存サイズが大きいものを指す。
描画速度	低速型	ティッカー、ウェブコンテンツアニメーション、全画面遷移（移動やフェード）などのレンダリングにおいて、誰が見てもコマ落ちしているのが良く分かるレベル。 静止画しか表示しないケースにおいては、低速型で十分対応可能。
	中速型	ティッカー、ウェブコンテンツアニメーション、全画面遷移（移動やフェード）などのレンダリングにおいて、違和感がないレベル。
	高速型	ティッカー、ウェブコンテンツアニメーション、全画面遷移（移動やフェード）などのレンダリングにおいて、一般の人がコマ落ちに全く気づけないレベル。
並行処理	シングル型	同時並行処理に効果が無い端末を指す。静止画を表示するだけのシンプルな用途であればこれで十分といえる。
	マルチ対応型	同時並行処理をサポートする端末を指す。並行処理が発生しても、動画やアニメーションの再生を妨げることがないレベル。
ビデオ再生	少機能型	ブラウザーランタイムがサポートする解像度より小さいサイズのビデオであれば再生できるレベル。
	中機能型	ブラウザーランタイムがサポートする解像度のビデオを全画面で再生できるレベル。
	高機能型	ブラウザーランタイムがサポートする解像度のビデオを全画面で再生できるだけでなく、縮小または拡大表示、回転表示、移動表示したまま再生できるレベル。
オーディオ再生	少機能型	単にオーディオを再生できるレベル。
	高機能型	オーディオの生成、合成、再生、エフェクトなどに対応したレベル。
再生開始遅延	低速型	再生開始コマンドを送ってから実際に再生が開始されるまでの遅延がある程度以上存在するもの。
	高速型	再生開始コマンドを送ってから実際に再生が開始されるまでの遅延が小さいもの。

3 解像度

解像度の分類と評価基準は、下表のとおりとする。

性能要件

分類	評価基準
小密度型	ブラウザーランタイムのビューポート解像度が画素数換算で 720P 相当（1280 × 720 ピクセル）以下の解像度のものを指す。
中密度型	ブラウザーランタイムのビューポート解像度が、画素数換算で 720P 相当（1280 × 720 ピクセル）より大きく 1080P 相当（1920 x 1080 ピクセル）以下のものを指す。
高密度型	ブラウザーランタイムのビューポート解像度が、画素数換算で 1080P 相当（1920 x 1080 ピクセル）を超えるものを指す。

ブラウザーランタイムのビューポート解像度は、ブラウザーに割り当てられるビデオメモリーのサイズに依存する。また、解像度を高くするには、ブラウザーランタイムが GPU アクセラレーションをサポートしないと、期待通りのパフォーマンスを上げることができない。

本項目は、ブラウザーランタイムのビューポート解像度の大きさによって分類する。すべての端末が、本分類のいずれかに属する。

3.1 小密度型

離れた場所から見ることを前提にした旧来型のサイネージを指す。小さな文字を表示するコンテンツには向かない。具体的には、ブラウザーランタイムのビューポート解像度が画素数換算で 720P 相当（1280 × 720 ピクセル）以下の解像度を指す。この HD 相当の解像度は、一般的なサイネージの用途として、もっとも普及している解像度であり、ほとんどのライトウェイトのデジタルサイネージの用途として、十分なサイズである。

3.2 中密度型

至近距離から見られることを前提に、小さな文字（映画の字幕より小さい文字）でも明瞭に読めるレベルのサイネージを指す。また、パネルのサイズが小さければ、ブランディングを重視したコンテンツにも使われる。具体的には、ブラウザーランタイムのビューポート解像度が、画素数換算で 1280x720 を超えて 1080P(フル HD)相当（1920 × 1080 ピクセル）以下のものを指す。

中密度型は、小密度型と比べて、詳細なピクセル表現が可能となり、とりわけ、近い距離なら小さな文字でも明瞭に読み取れる。フロア案内などのメッセージボードにも活用が可能である。また、42 インチ程度のパネルであれば、ブランディングを重視したコンテンツにも十二分に活用できる。

中密度型は、直近の Web-based Signage では必須の機能となっていくと考えられる。

3.3 高密度型

ブランディングやリアリティを重視したコンテンツを表示できるレベルのサイネージを指す。具体的には、ブラウザーランタイムのビューポート解像度が、画素数換算でフル HD 相当（1920 × 1080 ピクセル）を超えるものを指す。

高密度型は、大型パネルを使いつつも、粗が目立たない。また、パネルのサイズが小さければ、本物と見間違うほどのリアリティを再現できる。既存のハイエンドのデジタルサイネージシステムと同等の品質を提供する。

4 ストレージ

ストレージの分類と評価基準は、下表のとおりとする。

性能要件

分類	評価基準
キャッシュ型	コンテンツとなる画像ファイルや動画ファイルを保存する機能を有しない、または、保存機能は有するが、ブラウザーランタイムが W3C で規定された Indexed Database API※1 または File API: Directories and System※2 のいずれをも実装していない端末を指す。
蓄積型	コンテンツとなる画像ファイルや動画ファイルを保存する機能を有し、かつ、ブラウザーランタイムが W3C で規定された Indexed Database API※1 または File API: Directories and System※2 のいずれかを実装している端末を指すが、そのうち、保存サイズが 1024MB 未満のものを指す。
大容量蓄積型	コンテンツとなる画像ファイルや動画ファイルを保存する機能を有し、かつ、ブラウザーランタイムが W3C で規定された Indexed Database API※1 または File API: Directories and System※2 のいずれかを実装している端末を指すが、そのうち、保存サイズが 1024MB 以上のものを指す。

端末がストレージをサポートするかどうかによって、ネットワークトラフィック、端末起動からコンテンツ再生までの時間、安定的な長時間運転などのパフォーマンスに影響する。

本項目は、ストレージの有無、また、蓄積可能な容量によって分類する。すべての端末が、本分類のいずれかに属する。

なお、端末のストレージに事前蓄積してから再生するには、HTML5(Javascript 含む)の範囲外の手法で実現することもできる。本要求条件文書では HTML5 ベースの方法を対象としているため、この項目もそれを用いたものとなっている。

4.1 キャッシュ型

キャッシュ型は、ストレージを持たない、または、ストレージを有するものの、ブラウザーランタイムが W3C で規定された Indexed Database API※1 および File API: Directories and System※2 のいずれも実装していない端末を指す。

キャッシュ型は、端末を起動するたびにコンテンツのダウンロードが必要となり、コンテンツ再生までに時間がかかる。ただし、ブラウザーキャッシュによって、次回起動時におけるダウンロードデータ量は抑えられる可能性があるが、本分類では、ブラウザーキャッシュの有無は考慮しない。なお、ここで言うブラウザーキャッシュとは、Application Cache ではないので注意して欲しい。

また、ネットワークの状況によっては動画の再生が不安定になるなどのデメリットが想定される。一方、ストレージが無い分だけ、端末のハードウェアコストが抑えられる。また、コンテンツ配信サーバーと端末との間のネットワーク環境が良好であれば（オンプレミス型配信環境など）、さほど問題にならない。

4.2 蓄積型

蓄積型は、ストレージを持ち、かつ、ブラウザーランタイムが W3C で規定された Indexed Database API※1 または File API: Directories and System※2 のいずれかを実装している端末を指す。

蓄積型は、表示するコンテンツをストレージに蓄積し、コンテンツ再生時には、蓄積されたコンテンツを使うため、ネットワーク環境に依存しない安定再生が期待できる。また、コンテンツを都度ダウンロードする必要がないため、コンテンツが更新されていない限り、端末を起動してからコンテンツの再生までの時間が大幅に短縮される。

4.3 大容量蓄積型

大容量蓄積型は、蓄積型のうち、蓄積容量が 1024MB 以上のものを指す。

※1 W3C Indexed Database API

http://www.w3.org/TR/IndexedDB/

ストレージ型においては、Indexed Database API だけでなく、Blob オブジェクトの保存、および、File API 仕様で規定された Blob URL の生成（createObjectURL()メソッド）もサポートしなければいけない。

http://www.w3.org/TR/FileAPI/#dfn-createObjectURL

※2 W3C File API: Directories and System

http://dev.w3.org/2009/dap/file-system/file-dir-sys.html

2016 年 3 月現在、W3C File API: Directories and System は廃止となり、新たに File System API が考案されている。

http://w3c.github.io/filesystem-api/

ただし、まだ公開版草案になっておらず、ブラウザーの実装もないことから、しばらくは、File API: Directories and System が使われることになる。将来的には、File System API に置き換えられることになる。

5 描画速度

描画速度の分類と評価基準は、下表のとおりとする。

性能要件

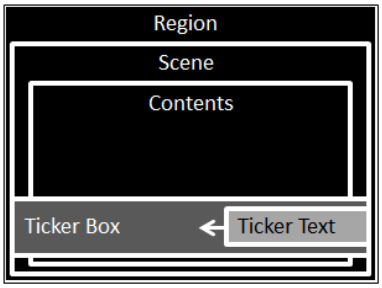
分類	評価基準
低速型	CSS transform / transition を利用して、ティッカーテキスト(画像ではなくフォントで実現し、高さが画面高の 10%)の水平移動をした場合及び静止画像（全画面）の水平移動を行った場合いずれかで、15 秒間でのコマ落ち(フレーム落ち)が合計 10 フレームを超えるものとする。
中速型	CSS transform / transition を利用して、ティッカーテキスト(画像ではなくフォントで実現し、高さが画面高の 10%)の水平移動をした場合及び静止画像（全画面）の水平移動を行った場合のいずれかで、15 秒間でのコマ落ち(フレーム落ち)の合計が 3 フレームを超えて、両方の場合で、10 フレーム以下となるものとする。
高速型	CSS transform / transition を利用して、ティッカーテキスト(画像ではなくフォントで実現し、高さが画面高の 10%)の水平移動をした場合及び静止画像（全画面）の水平移動を行った場合の両方で、15 秒間でのコマ落ち(フレーム落ち)が合計 3 フレーム以下となるものとする。

コンテンツアニメーションにおいてコマ落ち(フレーム落ち)は、パフォーマンスを測る上で、重要な指標となる。本項目は、コマ落ち(フレーム落ち)を基準に分類する。すべての端末が、本分類のいずれかに属する。

本評価は、以下の検証を総合的に判断して分類分けを行うこととする。

ティッカー

ティッカーはシーン領域上に用意されたティッカー領域（Ticker Box）の中を、ティッカーテキスト（Ticker Text）が右から左に人が読める速度で移動する。



パネル横置きを前提とし、全画面静止画コンテンツをシーン領域に表示した状態で、高さが全体の10%を占めるティッカー領域内で、ティッカーテキストが一定の時間をかけて右端から左端に到達する速度で移動するものとする。この状況で、ティッカーの動きを評価する。使用するティッカーテキストは、画像ではなくフォントで実現することとし、また十分な長さを有することとする。

遷移効果の検証には、CSS Transform と CSS Transition を採用する。

CSS の例
遷移前：transform: translateX(1920px); transition: transform 30s linear 0s;
遷移時：transform: translateX(-3840px);
※ 遷移前の translateX()の引数と transform の duration は画面横幅とテキストの幅を考慮して調整すること

画面遷移

シーン切り替えに使う遷移効果を検証材料とする。本評価においては、ブラウザーランタイムのビューポート解像度と同サイズの画像を使い、スライドの画面遷移を検証することとする。実際のサイネージコンテンツでは、1 秒以下の遷移を使うが、ここではパフォーマンス評価を目的とするため、計測可能な秒数で評価する。画面遷移にはスライドの他にフェードインやスケールもある。以下に事例を示す。

遷移効果の検証には、CSS Transform と CSS Transition を採用する。

フェードイン

CSS の例
遷移前：opacity: 0; transition: opacity 5s linear 0s;
遷移時：opacity: 1;

スライド

CSS の例
遷移前：transform: translateX(-1920px); transition: transform 5s linear 0s;
遷移時：transform: translateX(0px);
※ 遷移前の translateX()の引数は画面横幅に合わせること

スケール

CSS の例

遷移前：transform: scale(0); transition: transform 5s linear 0s;
遷移時：transform: scale(1)

5.1 低速型

いずれの評価においても、誰が見てもコマ落ちが気になるレベル。ティッカーは読みづらい。

5.2 中速型

いずれの評価においても、実用に支障がないレベルだが、よく見れば、一般の人でもコマ落ちが認識できるレベル。

5.3 高速型

いずれの評価においても、期待通りの表現が再現され、一般の人ではコマ落ちを認識することができないレベル。

6 並行処理

並行処理の分類と評価基準は、下表のとおりとする。

性能要件

分類	評価基準
シングル型	同時並行処理に効果が無い端末を指す。静止画を表示するだけのシンプルな用途であればこれで十分といえる。具体的には、ブラウザーランタイムが Web Workers を実装していない、または実装していたとしても、ワーカースレッド 2 個を使用して分散処理した場合に、1 個での処理に比べ、トータルの処理時間が短縮しないものを指す。
マルチ対応型	同時並行処理をサポートする端末を指す。並行処理が発生しても、動画やアニメーションの再生を妨げることがないレベル。具体的には、ブラウザーランタイムが Web Workers を実装しており、ワーカースレッド 2 個を使用して分散処理した場合に、1 個での処理に比べ、トータルの処理時間が短縮するものを指す。

Web-based Signage 用 JS プレーヤーは、プレイリストに基づいたメディア表示や再生だけではなく、平行してさまざまな処理を行う。また、マルチリージョンに対応した JS プレーヤーでは、複数の画面を分割表示しつつ、それぞれを独立して再生する必要がある。

この場合、レンダリングを伴わない処理をバックグラウンドで実行することで、メディア表示や再生を妨げることなく、さまざまな並行処理が可能となる。

本項目は、スレッドによる分割処理によって、トータルの処理時間の短縮が認められるかどうかで判断する。すべての端末が、本分類のいずれかに属する。

6.1 シングル型

シングル型は、ウェブランタイムが Web Workers を実装していない、または、実装しているが、スレッド処理をしても効果があまり認められないレベルを指す。一般的に、CPU がシングルコアの場合はこれに相当する。

6.2 マルチ対応型

マルチ対応型は、ブラウザーランタイムが Web Workers を実装しており、2 つ以上のスレッドを同時に処理することが可能であり、トータルの処理速度の短縮が認められるレベルを指す。一般的に、CPU がマルチコアの場合はこれに相当する。

マルチ対応型に属するためには、1 つのワーカースレッドでの処理時間に対して、2 つのワーカースレッドで分散処理した時にトータルの処理時間の短縮が認められることを条件とする。

7 ビデオ再生

ビデオ再生の分類と評価基準は、下表のとおりとする。

性能要件

分類	評価基準
少機能型	次を満たすこと。（1）mp4(H.264/AAC/MP4)もしくは webm(VP8/Vorbis/WebM)の少なくともいずれか一方をサポートすること。（2）Javascript から video 要素のオブジェクトの play()メソッドを実行したら、ビデオの再生が開始すること。 ビデオ解像度に関しては、ブラウザーランタイムがサポートする解像度のビデオを全画面で再生できなくてもよいとする。
中機能型	上記（1）（2）の条件に加え、ブラウザーランタイムがサポートする解像度の 30fps 以上のフレームレートのビデオをコマ落ちなく全画面で再生できること。
高機能型	上記（1）（2）の条件に加え、ブラウザーランタイムがサポートする解像度の 30fps 以上のフレームレートのビデオをコマ落ちなく全画面で再生でき、かつ、縮小または拡大後の再生、回転後の再生、移動後の再生（具体的には CSS Transforms を video 要素に適用しての再生）ができるレベル。

ビデオを再生するためには、それなりの CPU パワーと GPU パワーを要求する。特にブラウザーランタイムがサポートする解像度のビデオを全画面で再生するためには、ブラウザーランタイムがハードウェアデコードおよび GPU アクセラレーションに対応している必要がある。この差が、ビデオ再生のパフォーマンスに大きく影響を及ぼす。

ビデオ再生の分類に属するためには、以下の要件をすべて満たす必要がある。

1. 以下のビデオタイプのうち最低でも一つをサポートすること

MIME-Type	ビデオコーデック	オーディオコーデック	コンテナ
video/mp4	H.264	AAC	MP4
video/webm	VP8	Vorbis	WebM

2. 自動再生をサポートすること

   近年のスマートフォンやタブレット向け OS のブラウザーでは、ユーザーの操作なしに、JavaScript からビデオの自動再生を禁止しているものがある。本分類に属するためには、この制限を設けてはいけない。

   実際には、JavaScript から video 要素のオブジェクトの play()メソッドを実行したら、ビデオの再生が開始される必要がある。

たとえ、端末がビデオ再生をサポートしていたとしても、以上の要件を満たさない場合は、ビデオ再生の分類には属さない。

7.1 少機能型

少機能型は、主に、ソフトウェアデコードにしか対応していないブラウザーランタイムが該当する。CPU のみでデコード及びレンダリングを行うため、全画面のなめらかな再生は期待できないが、解像度が小さければ問題なく再生できる。

少機能型は、HTML5 仕様で規定された video 要素を使うことを前提としている。ただし、HTML 上にマークアップされた video 要素を再生するのではなく、すべて JavaScript から video 要素の生成と再生開始が行えなければいけない。

7.2 中機能型

中機能型は、前述の少機能型に加え、サイネージとしては最も一般的な用途を想定し、全画面のビデオ再生を必須とする。小さい解像度を引き伸ばして全画面表示する場合は、中機能型に属さない。少なくとも、ブラウザーランタイムがサポートする解像度と同じ解像度のビデオを拡大することなく全画面で滑らかに再生できるレベルを指す。30fps でエンコードされたビデオデータを、コマ落ちすることなく再生できなければいけない。

7.3 高機能型

高機能型は、中機能型をサポートするのはもちろんのこと、そのビデオを縮小または拡大表示、回転表示、移動表示しつつ、滑らかに再生できるレベルを指す。実際には、CSS Transform※を video 要素に適用しても、期待通りの変形を保ったまま再生できるレベルを指す。

※ CSS Transform

http://www.w3.org/TR/css3-transforms/

8 オーディオ再生

オーディオ再生の分類と評価基準は、下表のとおりとする。

性能要件

分類	評価基準
少機能型	次を満たすこと。（1）mp3(MP3/MP3)もしくは mp4(AAC/M4A), webm(Vorbis/WebM), ogg(Vorbis/Ogg)の少なくとも一種類のオーディオ再生をサポートすること。（2）Javascript から audio 要素のオブジェクトの play()メソッドを実行したら、オーディオの再生が開始すること。
高機能型	上記の条件に加え、Web Audio API の各種インターフェースの利用により、オーディオの生成、合成、エフェクトなどに対応したレベル。具体的には、少なくとも次のインターフェースをサポートすること。AudioContext, AudioNode, AudioDestinationNode, AudioBuffer, AudioBufferSourceNode, MediaElementAudioSourceNode, AudioParam, GainNode, DelayNode, OscillatorNode。

オーディオ再生の分類に属するためには、以下の要件をすべて満たす必要がある。

1. 以下のオーディオタイプのうち最低でも一つをサポートすること

MIME-Type	オーディオコーデック	コンテナ
audio/mp3	MP3	MP3
audio/mp4	AAC	M4A
audio/webm	Vorbis	WebM
audio/ogg	Vorbis	Ogg

2. 自動再生をサポートすること

   近年のスマートフォンやタブレット向け OS のブラウザーでは、ユーザーの操作なしに、JavaScript からオーディオの自動再生を禁止しているものがある。本分類に属するためには、この制限を設けてはいけない。

   実際には、JavaScript から audio 要素のオブジェクトの play()メソッドを実行したら、オーディオの再生が開始される必要がある。

たとえ、端末がオーディオ再生をサポートしていたとしても、以上の要件を満たさない場合は、オーディオ再生の分類には属さない。

8.1 少機能型

少機能型は、HTML5 仕様で規定された audio 要素を使うことを前提としている。ただし、HTML 上にマークアップされた audio 要素を再生するのではなく、すべて JavaScript から audio 要素の生成と再生開始が行えなければいけない。

8.2 高機能型

高機能型は、HTML5 仕様で規定された audio 要素を使ったオーディオを再生できるだけでなく、W3C Web Audio API を使ったオーディオの生成、合成、エフェクトなどに対応したレベルを指す。

Web Audio API を使うことで、音源をスクリプトから生成することでダウンロードデータの低減に寄与するだけでなく、複数の音源を合成することで、多彩なオーディオ表現が可能となる。

Web based Signage では、Web Audio API がカバーする高度なオーディオシンセサイジングのすべてを必要としない。基本的には、以下のユースケースをカバーできれば良い。

- ✓ audio 要素にセットされたオーディオを Web Audio API でコントロールする
- ✓ 複数の音源を同時に再生する
- ✓ 個々の音源に音量調整と遅延を適用する
- ✓ サイン波を指定することで単調な音源をスクリプトから生成する

高機能型に分類されるためには、上記のユースケースを実現する必要がある。少なくとも以下のインタフェースをサポートすれば、実現可能である。

- ✓ AudioContext
- ✓ AudioNode
- ✓ AudioDestinationNode
- ✓ AudioBuffer
- ✓ AudioBufferSourceNode
- ✓ MediaElementAudioSourceNode
- ✓ AudioParam
- ✓ GainNode
- ✓ DelayNode
- ✓ OscillatorNode

9 再生開始遅延

ビデオ・オーディオ再生時の遅延に関する性能要件は、下表のとおりとする。ビデオ・オーディオそれぞれについて評価する。

性能要件

分類	評価基準
低速型	再生開始コマンドを送ってから実際に再生が開始されるまでの遅延が、ビデオ・オーディオのいずれか一方もしくは両方において 0.5 秒以上のもの。
高速型	再生開始コマンドを送ってから実際に再生が開始されるまでの遅延が、ビデオ・オーディオ共に 0.5 秒未満のもの。

本評価は以下の手順で実施することとする。下記は video について記すが、audio も同様。

1. JavaScript から video 要素（HTMLVideoElement オブジェクト）を生成する。
2. 生成した HTMLVideoElement オブジェクトの preload プロパティに"auto"をセットする。
3. 生成した HTMLVideoElement オブジェクトに timeupdate イベントのリスナーをセットする。
4. 生成した HTMLVideoElement オブジェクトの src プロパティにビデオファイルの URL をセットする。
5. 生成した video 要素（HTMLVideoElement オブジェクト）をドキュメントに挿入する。
6. 数秒経過後に、HTMLVideoElement オブジェクトの play()メソッドを実行する。
7. play()メソッドを呼び出してから、最初に生成した timeupdate イベントが発生するまでの時間を、本評価の結果とする。

一般的なブラウザーでは、HTML 上に video 要素や audio 要素を生成し、src 属性にビデオ・オーディオファイルの URL をセットした時点で、再生開始位置から特定の範囲のビデオデータ・オーディオデータをダウンロードしデコード済みの状態でスタンバイする。これによって、いつでもすぐに再生できるようにすることができる（意図的に video 要素・audio 要素の preload 属性に"none"を指定した場合は除く）。近年のスマートフォンやタブレット向け OS のブラウザーでは、プリロードを禁止しているものがあるが、こういった環境では、高速型の要件を満たすことは難しいといえる。

多くのブラウザーでは、ビデオが再生されると timeupdate イベントは 250 ミリ秒ごとに発生する。そのため、play()メソッドを呼び出してから速やかにビデオが再生される環境であれば、500 ミリ秒（0.5 秒）の間に少なくとも 1 回は timeupdate イベントが発生するはずである。

play()メソッドを呼び出すまでに、十分なプリロード分のメディアデータがロードできるよう、本評価においては、十分なネットワーク帯域と、サーバーレスポンスを確保すること。

資料2

Web-based Signage 端末
性能テスト仕様

第 1 版

2016 年 3 月

作成： Web プラットフォーム性能ベンチマーク検討会

発行： 慶應義塾大学 SFC 研究所 先端ウェブアーキテクチャ・ラボ

目次

1.　本文書について

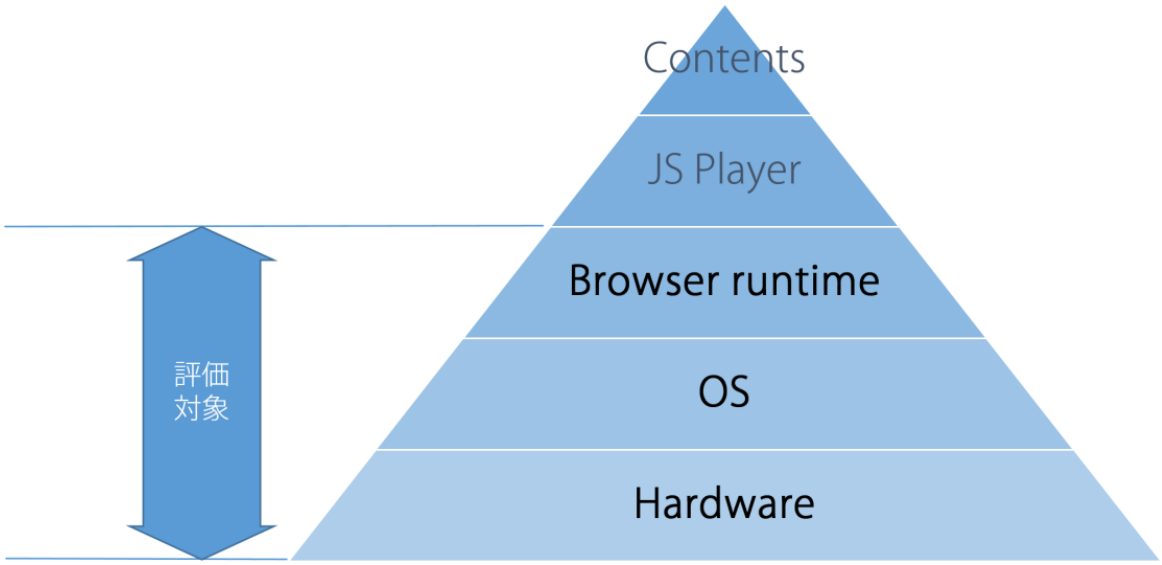
本文書は、「Web-based Signage 端末性能要件」文書に記載された、ウェブ(HTML5)ベースサイネージ端末に対する各性能要件について、評価試験を実施するためのテスト仕様である。性能要件の計 7 項目（解像度、ストレージ、描画速度、並行処理、ビデオ再生、オーディオ再生、再生開始遅延)それぞれに対応して、2 章の各節に試験内容を記載している。

性能評価試験としては、一般に、出来る限り自動化されることが望ましい。しかしながら、対象が PC ではなく組込系の端末であるということを前提とすると、目視など手動に頼らざるを得ないケースも多いと考えられる。したがって現時点では本文書でもそうした手動による試験手順を記載している項目もある。今後、実装ブラウザの進化等により、試験手順は改訂されていくことになろう。

1.1.　性能評価テストの範囲

性能評価の対象は、下記構成要素のうち、ハードウェア、オペレーティングシステム、ブラウザーランタイムである。

ただし、本性能評価は、構成要素を個別に評価するのではなく、これらを組み合わせた状態、つまり、実際の端末として総合的に評価する。



1.2. 性能評価の考え方

端末の性能の高低を単純に評価するのではなく、評価項目ごとの性能レベルを端末用途に応じた適合用途として分類し、その評価を実施することとする。

1.3. 本文書の利用シーン

本文書は、次のようなテストでの利用を想定している。

- 端末メーカーによる社内でのテスト
- 端末メーカーの依頼による第三者検証試験
- ロケーションオーナーもしくはサイネージシステム構築受託業者による採用候補端末のテスト（第三者委託も含む）
- その他

本文書の試験項目を実施することで、「Web-based Signage 端末性能要件」文書の規格に準拠していることを確認し、本文書で定義する各実装タイプ(Appendix 参照)に分類されることを認証することができる[1]。

1 ただし、プラットフォームとしての H/W とブラウザのすべての組合わせにおいてのパフォーマンスや動作を約束するものではない。

2. Web-based Signage 端末テスト仕様

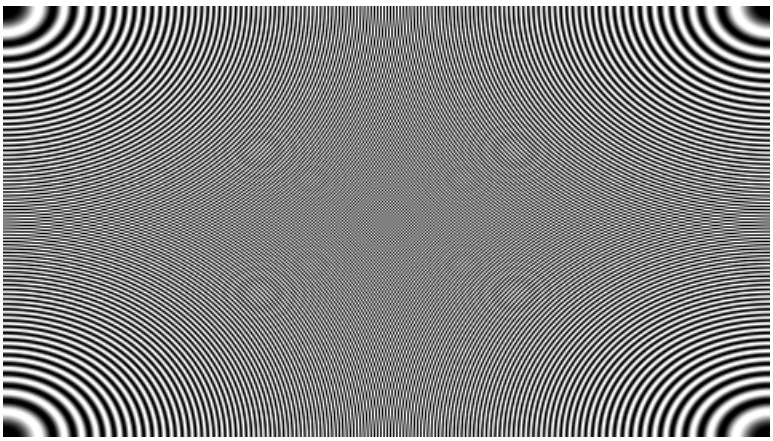
2.1. 解像度

ブラウザーランタイムのビューポート解像度の大きさによって 3 種(小密度型、中密度型、高密度型)に分類される。

機器が実装している分類(型)の条件を満たしていることを確認する。

2.1.1. 試験手順

申告値の解像度のゾーンプレート静止画（以下テストチャートという。下記に例を示すが、解像度が確認可能なものであればこれに限らない）を表示し、対応するブラウザーランタイムのビューポート解像度を確認する。例えば、目視により確認する。



2.1.2. 前提条件

当該テストチャートが読み込めること。

2.1.3. 実装条件/期待値

当該ゾーンプレート画の最高周波数部分で白黒の分離が確認できれば申告通りの解像度となる。最高周波数まで行かない場合、白黒分離が確認できる最大の周波数を読み取る。

その結果に基づき、小密度型（1280 × 720 ピクセル以下）、中密度型（1920 x 1080 ピクセル以下）、高密度型（1920 x 1080 ピクセルを超えるもの）に分類する。

2.1.4. 備考

特になし。

2.2. ストレージ

ストレージの有無、また、蓄積可能な容量によって 3 種(キャッシュ型、蓄積型、大容量蓄積型)に分類される。

機器が実装している分類(型)の条件を満たしていることを確認する。

2.2.1. 試験手順

合計 1024MB となる画像ファイルを準備し、Indexed Database API または File API: Directories and System を利用して端末に保存を試みる。

2.2.2. 前提条件

特になし。

2.2.3. 実装条件/期待値

保存機能が全くないもしくは Indexed Database API と File API: Directories and System のいずれも実装されていない端末はキャッシュ型とする。

保存可能なファイルサイズが 1024MB 未満の端末を蓄積型とする。

保存可能なファイルサイズが 1024MB 以上の端末を大容量蓄積型とする。

2.2.4. 備考

特になし。

2.3. 描画速度

ティッカーやアニメーションの描画時の速度性能（フレームレート）を測定する。

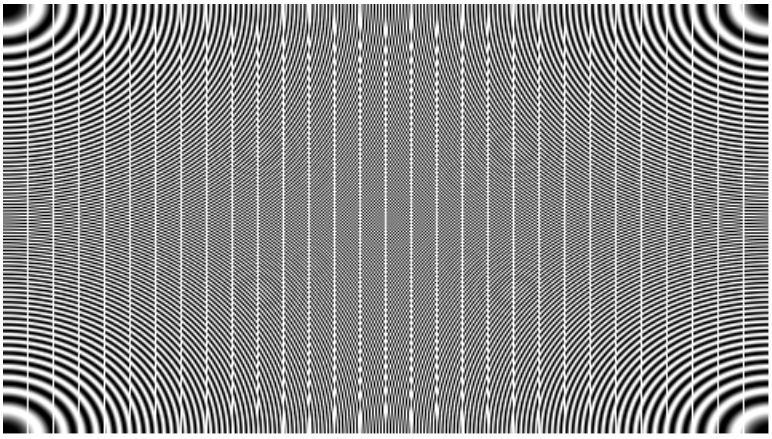
2.3.1. 試験手順

ティッカー

様々な文字を含むティッカー(高さ:画面高の 10%)を準備する（画像ではなくフォントで実現する）。またティッカー画像を水平スクロールさせる際の背景画像としては、細い等間隔の縦縞模様を有するものとする(例:下図参照)。CSS transform / transition を利用して、ティッカーテキストの水平移動を行う。その速度は、1 フレームあたり文字列が縞 1 本分進む速度とする。実際に端末上で画像表示させ、フレームレートを測定する。例えば、表示ディスプレイをビデオカメラで録画しその変化量から測定する。

全画面移動

等間隔の縦縞模様の重畳された静止画像(全画面)を準備する(例:下図参照)。CSS transform / transition を利用して、この画像の水平移動を行う。その速度は、1 フレームあたり縞 1 本分進む速度とする。実際に端末上で画像表示させ、コマ落ちの程度を測定する。例えば、表示ディスプレイをビデオカメラで録画しそのスロー再生の目視から測定する。



2.3.2. 前提条件

特になし。

2.3.3. 実装条件/期待値

ティッカー・全画面移動のいずれかで 15 秒間の再生中のコマ落ち(フレーム落ち)が合計 10 フレームを超える時、低速型と判定する。

ティッカー・全画面移動のいずれかで 15 秒間の再生中のコマ落ち(フレーム落ち)の合計が 3 フレームを超えて、両方で、10 フレーム以下となる時、中速型と判定する。

ティッカー・全画面移動の両方で 15 秒間の再生中のコマ落ち(フレーム落ち)が合計 3 フレーム以下の時、高速型と判定する。

2.3.4. 備考

特になし。

2.4. 並行処理

端末の同時並行処理性能を評価する。

2.4.1. 試験手順

ある計算処理について、Web Workers を使用するコードを作成しておく。ワーカースレッドを 2 個使用した場合と 1 個使用した場合の処理時間を比較する。

2.4.2. 前提条件

特になし。

2.4.3. 実装条件/期待値

ワーカースレッドを 2 個使用しても処理時間が短縮しない場合、シングル型と判定する。

ワーカースレッドを 2 個使用した場合に処理時間が短縮する場合、マルチ対応型と判定する。

2.4.4. 備考

特になし。

2.5. ビデオ再生

映像の再生性能を評価する。

2.5.1. 試験手順

1 ブラウザーランタイムのビューポート解像度(全画面)のビデオ映像(30fps)を準備しておく。各フレームにはフレーム番号をスーパーインポーズしておく。これを、サポート申告のあったビデオタイプ(mp4 もしくは webm)及びビットレートで、エンコードしておく。

2 JS プレーヤーで全画面でビデオ再生し、コマ落ちの有無を確認する。例えば、表示ディスプレイをビデオカメラで録画し、再生することでコマ落ちの有無を目視で調査する。

3 JS プレーヤーで CSS Transform を video 要素に適用し、縮小表示・拡大表示・回転表示・移動表示を行いビデオ再生し、コマ落ちの有無を確認する。例えば、表示ディスプレイをビデオカメラで録画し、再生することで、コマ落ちの有無を目視で調査する。具体的には CSS Transform にて次の数値を用い、遷移後の位置・大きさでビデオ再生する。縮小:0.5 倍、拡大：2 倍、回転：時計回りに 90 度、移動：右方向 100 画素・下方向 100 画素。

2.5.2. 前提条件

以下を満たすことが本評価の前提条件である。

次の 2 種類のビデオタイプのうち少なくともいずれか一方をサポートすること。

MIME-Type	ビデオコーデック	オーディオコーデック	コンテナ
video/mp4	H.264	AAC	MP4
video/webm	VP8	Vorbis	WebM

2. ユーザーの操作なしに自動再生をサポートすること。具体的には、JavaScript 上で video 要素の play()メソッドが実行されたら、ビデオの再生が開始されること。

2.5.3. 実施条件/期待値

試験手順②において、30fps のビデオがコマ落ちなく全画面で再生できる場合、中機能型(以上)と判定する。この際に再生したストリームのビットレートも記録する。

試験手順③においても 30fps のビデオがコマ落ちなく再生できる場合、高機能型と判定する。

上記の前提条件は満たすものの中機能型・高機能型と判定できないものは、少機能型とする。

2.5.4. 備考

特になし。

2.6. オーディオ再生

オーディオ再生性能を評価する。

2.6.1. 試験手順

1 適当なオーディオファイルを準備しておく。

2 JS プレーヤーで audio 要素の play()によりオーディオ再生し、動作を確認する。例えば、実際に聴いて確認する。

3 audio 要素にセットされたオーディオに対する下記の処理を、Web Audio API の次のインターフェースを用いてスクリプト上に実現する。

i. 【使用インターフェース】

- AudioContext
- AudioNode
- AudioDestinationNode
- AudioBuffer
- AudioBufferSourceNode
- MediaElementAudioSourceNode
- AudioParam
- GainNode
- DelayNode
- OscillatorNode

ii. 【実現する処理】

- 音源に異なる音量調整と遅延量調整を行って同時に再生する。
- スクリプト上で 1kHz サイン波音源を生成し再生する。
- オーディオファイルをサーバーから取り込み再生する。

4 前項の処理によるオーディオを JS プレーヤーで再生し、動作を確認する。例えば、実際に聴いて確認する。

2.6.2. 前提条件

以下を満たすことが本評価の前提条件である。

a) 次の 4 種類のオーディオタイプのうち少なくともいずれか 1 種類をサポートすること。

MIME-Type	オーディオコーデック	コンテナ
audio/mp3	MP3	MP3
audio/mp4	AAC	M4A
audio/webm	Vorbis	WebM
audio/ogg	Vorbis	Ogg

b) ユーザーの操作なしに自動再生をサポートすること。具体的には、JavaScript 上で audio 要素の play()メソッドが実行されたら、オーディオの再生が開始されること。

2.6.3. 実施条件/期待値

試験手順に記した Web Audio API のインターフェースのすべての正常動作が確認できた場合、高機能型と判定する。

前提条件は満たすもののいずれかのインターフェースが動作しない場合は、少機能型と判定する。

2.6.4. 備考

特になし。

2.7. 再生遅延時間

再生開始コマンド後、実際にオーディオ・ビデオが再生されるまでの遅延時間を評価する。

2.7.1. 試験手順

ビデオの再生遅延

1 JavaScript から video 要素（HTMLVideoElement オブジェクト）を生成する。
2 生成した HTMLVideoElement オブジェクトの preload プロパティに"auto"をセットする。
3 生成した HTMLVideoElement オブジェクトに timeupdate イベントのリスナーをセットする。
4 生成した HTMLVideoElement オブジェクトの src プロパティにビデオファイルの URL をセットする。
5 生成した video 要素（HTMLVideoElement オブジェクト）をドキュメントに挿入する。
6 プリロードが完了するのに十分な時間(数秒)の経過後に、HTMLVideoElement オブジェクトの play()メソッドを実行する。
7 play()メソッドを呼び出してから最初に生成した timeupdate イベントが発生するまでの時間を測定し、ビデオの再生遅延時間とする。

オーディオの再生遅延

1 JavaScript から audio 要素（HTMLAudioElement オブジェクト）を生成する。

2 生成した HTMLAudioElement オブジェクトの preload プロパティに"auto"をセットする。

3 生成した HTMLAudioElement オブジェクトに timeupdate イベントのリスナーをセットする。

4 生成した HTMLAudioElement オブジェクトの src プロパティにビデオファイルの URL をセットする。

5 生成した audio 要素（HTMLAudioElement オブジェクト）をドキュメントに挿入する。

6 プリロードが完了するのに十分な時間(数秒)の経過後に、HTMLAudioElement オブジェクトの play()メソッドを実行する。

7 play()メソッドを呼び出してから最初に生成した timeupdate イベントが発生するまでの時間を測定し、オーディオの再生遅延時間とする。

2.7.2. 前提条件

ビデオ再生またはオーディオ再生の性能が少なくとも中機能型以上に分類されていること。

2.7.3. 実施条件/期待値

play()(再生開始コマンド)実行後、実際に再生開始されるまでの遅延が、ビデオ・オーディオどちらかまたは両方とも 0.5 秒以上の場合、低速型と判定する。

同遅延がビデオ・オーディオ共に 0.5 秒未満の場合、高速型と判定する。

2.7.4. 備考

本評価においては、play()を呼び出す前にデータをプリロードする必要があるため、十分なネットワーク帯域とサーバーレスポンスを確保することが望ましい。

3. Appendix

3.1. Web-based Signage 端末の実装タイプとテスト適合要件

本性能評価分類及びその評価分類ごとの性能を軸とした適合分類に基づき、各端末の典型的な実装タイプが分類できる。以降、テスト適合要件の記号定義として、以下を用いる。

M　Mandatory: 必要要件

O　Option: オプション

3.1.1. 静止画・スライドショー型

例えば、小規模な小売店舗で店の情報などを表示する場合や企業オフィスで従業員向けの情報を伝える場合において、動画は多用せず、静止画のアニメーションやテキストを主とするケースを想定している。このような静止画・スライドショー型のテスト適合要件は以下のようになる。

(出典：DSC デジタルサイネージシステムガイドブック)

#	評価分類	適合分類	適用要件
1	解像度	小密度型	**M**
		中密度型	**O**
		高密度型	**O**
2	ストレージ	キャッシュ型	**-**
		蓄積型	**M**
		大容量蓄積型	**O**
3	描画速度	低速型	**M**
		中速型	**O**
		高速型	**O**

#	評価分類	適合分類	適用要件
4	並行処理	シングル型	**M**
		マルチ対応型	**O**
5	ビデオ再生	少機能型	**M**
		中機能型	**O**
		高機能型	**O**
6	オーディオ再生	少機能型	**M**
		高機能型	**O**
7	再生開始遅延	低速型	**O**
		高速型	**O**

3.1.2. 動画型

動画型では、静止画・スライドショー型に加え、ビデオ再生を多用したコンテンツにも対応するケースを想定している。例えば、イメージ映像や環境映像などを表示する場合が考えられる。このような動画型のテスト適合要件は以下のようになる。

#	評価分類	適合分類	適用要件
1	解像度	小密度型	**M**
		中密度型	**O**
		高密度型	**O**
2	ストレージ	キャッシュ型	**-**
		蓄積型	**-**
		大容量蓄積型	**M**
3	描画速度	低速型	**-**
		中速型	**M**
		高速型	**O**

#	評価分類	適合分類	適用要件
4	並行処理	シングル型	**-**
		マルチ対応型	**M**
5	ビデオ再生	少機能型	**-**
		中機能型	**M**
		高機能型	**O**
6	オーディオ再生	少機能型	**-**
		高機能型	**M**
7	再生開始遅延	低速型	**M**
		高速型	**O**

3.1.3. 複合型

複合型は、全評価項目で最高クラスに属し、様々な表現が可能となる。例えば、高品質なハイビジョン映像や高精細静止画などで構成されるコンテンツも表示するケースが考えられる。このような複合型のテスト適合要件は以下のようになる。

#	評価分類	適合分類	適用要件
1	解像度	小密度型	**-**
		中密度型	**-**
		高密度型	**M**
2	ストレージ	キャッシュ型	**-**
		蓄積型	**-**
		大容量蓄積型	**M**
3	描画速度	低速型	**-**
		中速型	**-**
		高速型	**M**

#	評価分類	適合分類	適用要件
4	並行処理	シングル型	**-**
		マルチ対応型	**M**
5	ビデオ再生	少機能型	**-**
		中機能型	**-**
		高機能型	**M**
6	オーディオ再生	少機能型	**-**
		高機能型	**M**
7	再生開始遅延	低速型	**-**
		高速型	**M**

3.2. Web-based Signage 端末性能テスト申請例

■基本情報

会社名	
住所	
所属・役職	
担当者氏名（ふりがな）	
電話番号/Fax 番号	
E-Mail	
製品名（型番）	
機器カテゴリー	

■機器の仕様

機器がサポートしている仕様（値）を記入してください。

解像度(ピクセル)	ブラウザーランタイムのビューポート解像度		× ピクセル
ストレージ(バイト)	コンテンツとなる画像・動画ファイルを保存する機能の有無。 保存する機能がある場合は保存サイズ		有 ・ 無 MB
描画速度(フレーム)	ティッカーと全画面移動の各場合の 15 秒間再生中のコマ落ちフレーム数		ティッカー： frames 全画面移動： frames
並行処理	同時並行処理のサポート有無		有 ・ 無
ビデオ再生	サポートするビデオタイプ	MIME-Type	
		ビデオコーデック	
		オーディオコーデック	
		コンテナ	
	自動再生サポートの有無		有 ・ 無
	全画面再生可能なフレームレート、ビットレート		fps, Mbps
	縮小・拡大・回転・移動後のビデオ再生機能		有 ・ 無
オーディオ	サポートするオーディオタイプ	MIME-Type	
		オーディオコーデック	
		コンテナ	
	自動再生サポートの有無		有 ・ 無
	オーディオイフェクト等の処理(2.6.1)への対応		有 ・ 無
再生開始遅延	再生開始コマンド後、実際に再生開始されるまでの時間		msec

3.3. Web-based Signage 端末性能テスト結果例

試験判定結果と 3.1 で述べた Web-based Signage の実装タイプにおける各適用要件とを照合することで、試験した機器の実装タイプを分類することもできる。下記の場合、複合型をベースに描画速度が中速型の実装タイプであると分類できる。

#	機能		申請値	試験判定	適用要件（複合型）
1	解像度	小密度型	-	-	-
		中密度型	-	-	-
		高密度型	**ピクセル	**Pass** **＊＊ピクセル**	**M**
2	ストレージ	キャッシュ型	-	-	-
		蓄積型	-	-	-
		大容量蓄積型	**MB	**Pass** **＊＊MB**	**M**
3	描画速度	低速型	-	-	-
		中速型	-	Pass **frames	-
		高速型	**frames	**Fail**	**M**
4	並行処理	シングル型	-	-	-
		マルチ対応型	対応	**Pass**	**M**
5	ビデオ再生	少機能型		-	-
		中機能型	-	-	-
		高機能型	**Mbps	**Pass** ****Mbps**	**M**
6	オーディオ	少機能型	-	-	-
		高機能型	対応	**Pass**	**M**
7	再生開始遅延	低速型	-	-	-
		高速型	300msec	Pass ＊＊msec	**M**

資料３

Web-based Signage
コンテンツ開発ガイドライン

第１版

2016 年 3 月

作成： Web プラットフォーム性能ベンチマーク検討会

発行： 慶應義塾大学 SFC 研究所 先端ウェブアーキテクチャ・ラボ

目次

1. コンテンツ開発ガイドラインの概要

本資料は、Web-based Signage 普及を目的に、コンテンツ制作者向けにハードウェア上でもっとも効果的に動作するコンテンツを制作するためのガイドラインを示す。

1.1. ガイドラインの範囲

Web-based Signage は、コンテンツを管理する CMS（コンテンツ登録やプレイリストなどを管理）、コンテンツを配信する CDN（配信クラウドとネットワーク）、そして、コンテンツを再生する端末から構成される。本ガイドラインは、これら Web-based Signage システム全体のうち、コンテンツ部分を対象とする。Web-based Signage システムは、以下の要素で構成される。

ハードウェア

物理的に端末を構成する構成要素を指す。具体的には、パネル、SoC（CPU, GPU, メモリなど）、入出力インタフェース（HDMI, USB, 電源など）で構成される。

オペレーティングシステム

Web-based Signage として端末ハードウェアを利用するためにインストールされる基本ソフトを指す。具体的には、Windows, Android, iOS, Linux, FreeBSD, Firefox OS などを指す。

ブラウザーランタイム

HTML, CSS, JavaScript などの Web 標準技術を動作させることができるアプリケーション基盤を指す。具体的にはウェブブラウザーを指すが、必ずしも、一般的なウェブブラウザーである必要はない。例えば、Android の WebView など、ランタイムとして用意された環境も含む。

JS プレーヤー

プレイリストに基づいてコンテンツを再生するためのアプリケーションを指す。Web-based Signage では、ブラウザーランタイム上で動作するウェブアプリケーションとして作られる。具体的には、HTML, CSS, JavaScript を駆使して開発される。

コンテンツ

コンテンツは、JS プレーヤー上で表示または実行され、サイネージ端末の前にいるユーザーに見えるものである。コンテンツは、単体画像、動画、または、ウェブアプリケーションとして制作される。中には、これらコンテンツを表示しつつ、音声を再生するものも存在する。

1.2. ガイドラインの方針

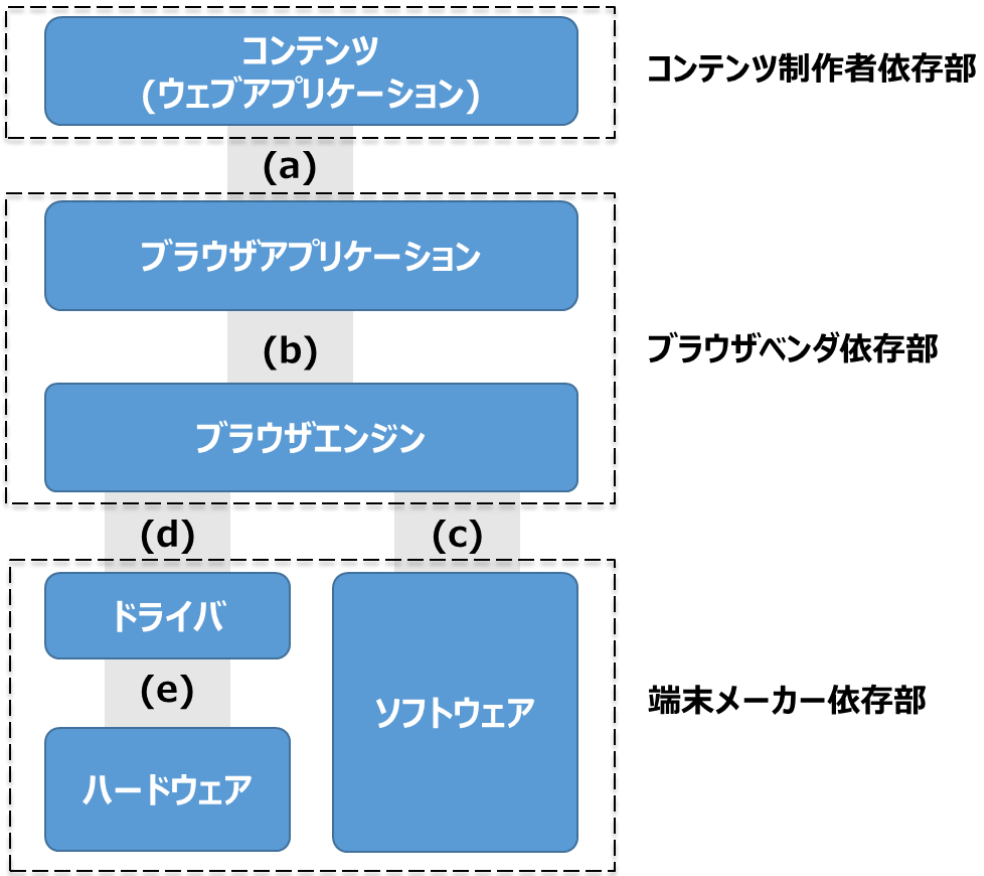
PC 等に比べ、RAM/ROM/CPU 等の資源が少ない組み込み機器上で動作するブラウザーランタイム、JS プレーヤーのパフォーマンスを引き出すための指針を示す。具体的なコード例を避け、コンテンツ制作者の創意工夫の余地を残しながら、原則として従うべきグッド/バッドノウハウを紹介する。

2. 開発をはじめる前に

組み込み機器向けにコンテンツ(ウェブアプリケーション)を制作する上で、表示対象となる端末・ブラウザの特性を把握することが非常に重要である。本章では、開発をはじめる前にどういった端末情報を把握すべきかについて記述する。

2.1. 開発対象の全体像と特性の把握

参考情報としてブラウザ搭載組み込み機器の概念図を以下に示す。この図の中の(a)以外については、コンテンツ制作者が関与できない部分のため、『Web-based Signage 試験結果表』を事前に入手し、表示対象となる端末・ブラウザの特性を把握することを推奨する。



(a) HTML コンテンツ

唯一コンテンツ制作者がコントロールできる部分。コンテンツ制作者は(b)~(e)の特性を考慮した企画・デザインを行うことが肝要である。『Web-based Signage 試験結果表』に記されている特性を理解した上でコンテンツの内容を検討し、表示対象となる端末・ブラウザに最適なコンテンツ設計・実装をおこなう必要がある。

(b)ブラウザ内部

「ブラウザ」は前述の図の様に、大きく分けてアプリケーション、エンジン、端末との結合部分と３つに分けることができる。ブラウザアプリケーションとは、主にブラウザエンジンの起動や終了、メニュー、ブックマークなどの UI 表示などのコンテンツ実行以外の付帯部分、さらには端末側からのキー入力やタイマーなどの OS やハードウェアからの割り込みをきっかけにしてブラウザエンジンに対して制御命令をだす部分である。ブラウザアプリケーションは、上述の様にコンテンツ以外の外部の入力をブラウザに伝える役割をしている。その関係で、同じブラウザエンジンを利用していても、ブラウザアプリケーション次第では異なる挙動となる。後述の(c)(d)による影響が大きい場合もあるが、表示対象となる端末に、特に気をつけるべき動作制限が公開情報としてないかを事前に調査しておくことを推奨する。なお、ブラウザベンダによっては PC 上で動作するシミュレータを用意していることもあり、必要に応じて入手・活用することを推奨する。

(c)ブラウザ移植部（ソフトウェア実装）

ブラウザエンジンの設計次第ではあるが、ハードウェアを利用できそうな機能は、ブラウザエンジンの外で処理を行えるように切り出されている場合がある。端末の仕様によるが、これらの切り出された機能がハードウェアに結合されている場合と、ソフトウェアで実装されている場合がある。『Web-based Signage 試験結果表』を参照することで、これらの特性がつかめてくると思うが、後述の GPU アクセラレーションが無効、もしくはほぼ効いていないような場合は、これらの機能はなるべく利用を控えることを推奨する。例えば、アニメーションなどの描画性能は GPU アクセラレーションが無効な場合は CPU やメモリバス速度に依存してくる。この場合、アニメーション、グラデーション、アルファ値の操作などが高負荷処理となる。

(d)ブラウザ移植部（ハードウェア結合）

GPU アクセラレーションが有効な場合は、ブロック要素のレイヤー化などが利用できる可能性があるため、より滑らかなアニメーションが期待できる。レイヤー化の詳細については、後述の「3．ノウハウ集」を参照とするが、レイヤー化を意識したコンテンツ制作をおこなう必要

があるため、企画・デザインの検討に入る前に表示対象の端末がGPUアクセラレーションに適応しているかどうか調査しておくことを推奨する。

(e)デバイスドライバーハードウェア間

(c)(d)にて記述したハードウェア・アクセラレーションが有効な場合においても、デバイスドライバとハードウェアの間の結合品質によっては、期待どおりのパフォーマンスが出ないことがある。表示対象の端末で、このような既知の問題が存在するかを企画・デザインの検討に入る前に調査しておくことを推奨する。

2.2. 開発プロセスについて

開発をはじめる前に対象となる端末を入手し、開発中のコンテンツを随時テストできる環境を構築することを強く推奨する。開発効率を優先して一般的なPCブラウザにて開発を進めることについては否定しないが、出来るだけ実際の端末上で動作確認をおこなうことを推奨する。実機での動作は、PCの動作と比べて動作速度や見え方などが異なる場合も多く、コンテンツ作成時から実機で確認していくことでコンテンツ作成の手戻りを防ぐことが出来る。たとえば、ブラウザエンジンの種類や端末毎にサポートする機能が微妙に異なっていることや、利用可能なメモリ量にも違いがある場合があり、端末上でのみエラーが発生することもある。また、動作速度の違いにより期待するUXが実現できないこともある。一般的なPCブラウザやブラウザベンダが提供するPCシミュレータ等を活用し、効率的に開発を進めつつ、定期的に対象となる端末上で動作確認をおこなうことが重要である。

3. ノウハウ集

3.1. 表記ルールについて

以降に記述する各ノウハウについて、そのノウハウにより良化が見込まれる項目や、逆に悪化する可能性のある項目(トレードオフ)を以下のタグ表記により示すものとする。

良化の可能性あり

悪化の可能性あり

項目：

[CPU] … CPU 負荷に影響するため CPU スペックが低い端末での効果が期待できる項目

[通信] … 通信負荷に影響するため、主にコンテンツ表示までの速度の改善が見込める

[メモリ] … メモリ使用量が減るため、メモリ搭載量の少ない端末でのメモリ不足回避が見込める

[描画] … 描画処理の負荷が軽減されるため、UX の改善に繋がる可能性がある

[フォント] … 文字描画処理の負荷に影響するため、文字を多用するコンテンツでの改善が見込める

[保守性] … 成果物の運用・メンテナンス時の手間の削減が見込める

表記例：

通信 メモリ

通信処理の高速化が見込めるが、トレードオフとしてメモリ使用量が増えるという意味。

3.2. コンテンツの構成と配置

3.2.1. jQuery 等巨大ライブラリの利用を避ける

CPU 通信 メモリ 描画

高度なアニメーション処理が必要とされる場合を除き、原則としてミニファイした状態で数百 KB 以上の容量をもつライブラリの利用を避ける。高度な用途向けに設計されたライブラリは一般的に処理負荷が高く、シンプルな処理に利用すると、スムーズなコンテンツ再生の妨げとなることが多い。また、巨大ライブラリを利用しないことで、通信処理や JS 解析処理の軽減により起動高速化やメモリ使用量の削減などの効果も期待できる。

3.2.2. JavaScript/CSS ファイルは結合し、ミニファイする

通信 保守性

JavaScript ファイルや CSS ファイルはリクエスト回数の削減、および処理系の負荷軽減を目的とし、ファイルを結合したのち、ミニファイ(難読化でも同様の効果)する。これらを HTML ファイル中に埋め込むことで、さらにリクエスト回数を削減するという手法もあるため、開発規模や保守性などを考慮した上で適切な手法を選定することを推奨する。

3.2.3. 部品利用する画像は CSS Sprite 等を用いて 1 ファイルとする

通信 メモリ 保守性

部品として利用する小さい画像については CSS Sprite(1 枚の画像の特定領域のみを padding, overflow: hidden 等で表示する技法)等を用いてできるだけ 1 ファイルにまとめる。これにより、リクエスト回数を削減することができる。ただし、利用頻度がそれほど高くない画像を含んでしまった場合にメモリ使用量が単体ファイル利用時よりも増えてしまう場合があるため、対象とする画像の選定は慎重におこなう必要がある。

3.3. コンテンツ記述時

3.3.1. 動的な DOM 生成時は DocumentFragment を利用する

CPU 描画

DOM ツリーの操作にはそれに対応するメモリ操作や再描画処理が伴うため、Native API の appendChild()を繰り返し呼び出して HTML を生成することや、jQuery の $.append() や $.appendTo()を繰り返し呼び出して HTML を生成することは避けた方がよい。代わりに DocumentFragment により仮想的な DOM ツリーを構築してから、DOM ツリーに appendChild することで再描画を最低限に抑制することができる。DocumentFragment が利用できない場合は String で出力する（innerHTML 等）などの手法でも実現できる。

3.3.2. CSS の gradient/box-shadow/border-radius を控えめにする

描画

大きな画像に対する CSS での飾り付け処理は、ブラウザーランタイムにとって非常に負荷の高い処理となるため基本的には避ける。動的にトランジションしていく場合など、特別の理由がない限り元の画像のほうで付与しておく。特に GPU アクセラレーションの有無により、大きくパフォーマンスが異なる可能性があるため、企画・デザイン検討段階で、端末性能を正しく把握するためにコンテンツのプロトタイプを用意し動作確認を実施することを推奨する。

3.3.3. フォントがキャッシュに収まるように文字スタイルを絞る

フォント

環境に依存するが、基本的にフォント描画は処理コストは高い。一度描画したフォントを使いまわし、キャッシュに残るように工夫する。特に文字列をアニメーションさせる場合など、頻繁にフォント描画が発生するようなコンテンツでは効果が得られる可能性がある。

3.3.4. リフローが与える影響を考慮する

CPU 描画

親要素に対する class / style の変更は、子孫要素へも影響があるため、再計算・再描画処理が大量に発生してしまう場合がある。特に深い階層をもつ要素ではパフォーマンスへの影響が甚大なため、コンテンツ制作者はリフロー発生を意識した設計・実装をすること。

3.3.5. マルチコア環境では Web Worker を活用する

CPU

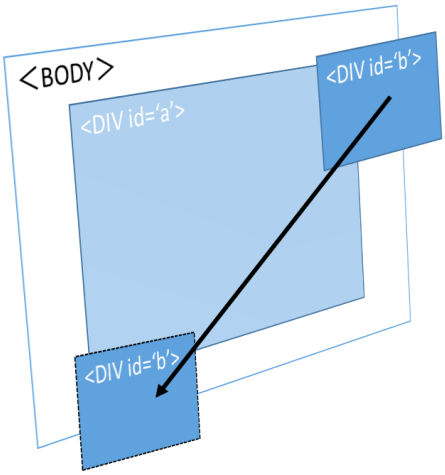
DOM 処理等の長大な JavaScript 処理を複数実行する必要がある場合かつ、性能要件における「マルチ対応型」に適合する環境であれば、積極的に Web Workers を活用する。

3.3.6. GPU の活用

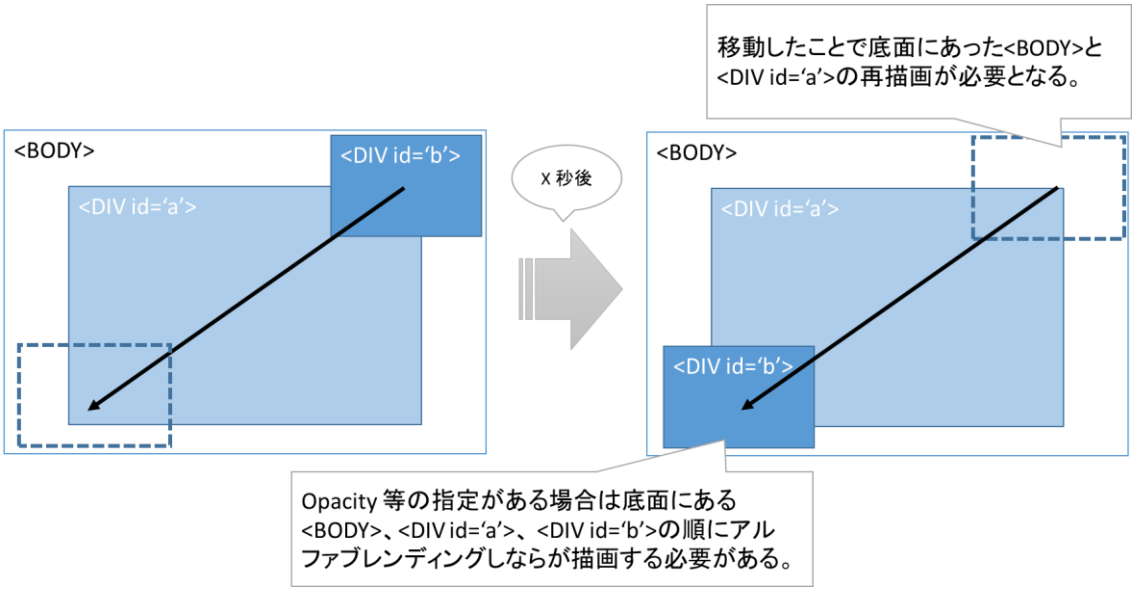
CPU 描画

ブラウザエンジンの種類やその移植方法によっても活用の仕方が異なるため一意に説明することは難しいため、ここでは Webkit 系のブラウザについて説明する。Webkit では Accelerated-Compositing と呼ばれる機能があり、この機能が有効化されたブラウザでは、指定したブロック要素を別レイヤー化することでアニメーションの高速化ができる可能性がある。Accelerated-Compositing はその名のとおり、「合成処理」を高速化するというものである。具体的には 1 枚の絵として生成していたコンテンツを複数のレイヤーに分けてレイヤー同士を GPU で合成(ブレンディング)することで高速化を図るというものである。

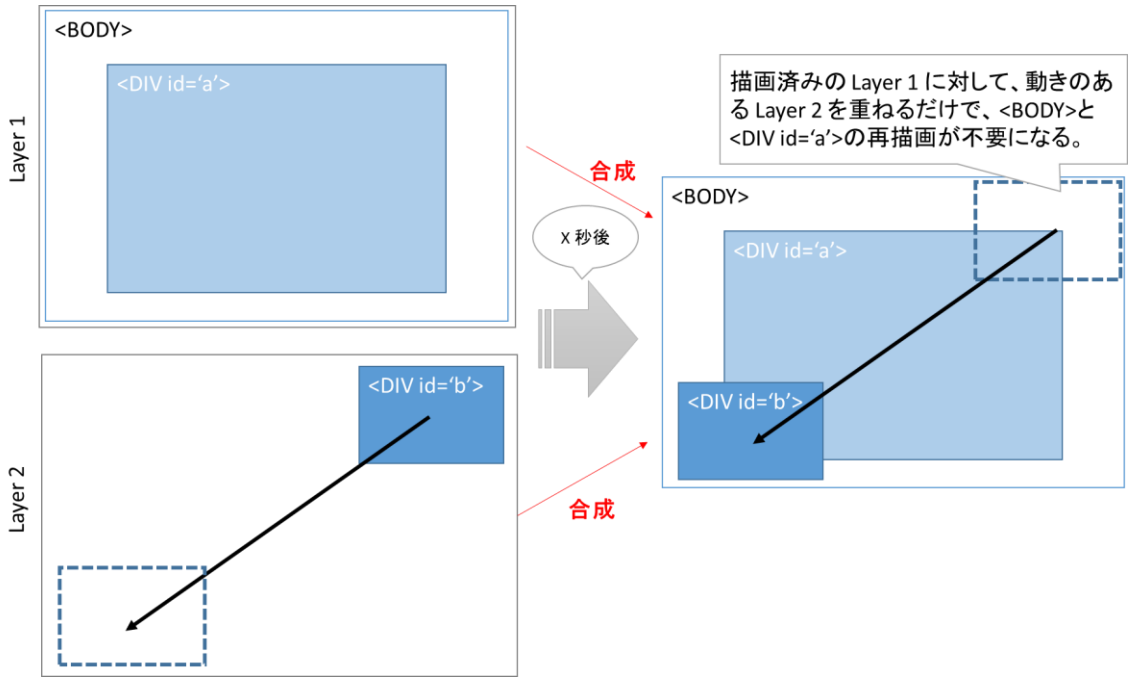
以下では <BODY><DIV id='a'>の要素の上で<DIV id='b'>がアニメーションする場合を例にレイヤー化の概念を説明する。



まず、レイヤー化ができない場合、以下の図のようにアニメーションが発生する度に、DOM ツリーを辿って何度も再描画する必要があり、描画処理に時間がかかってしまうためアニメーションのフレームレートが上がらない場合が多い。



一方でレイヤー化できる場合、<BODY><DIV id='a'>と<DIV id='b'>の描画レイヤーを分離し、それぞれのレイヤーを GPU で合成することで、再描画処理の軽減が見込める。



但し、レイヤーの多用は VRAM を著しく消費するため、端末によっては正常な描画ができなくなる場合があるため注意が必要である。加えて、レイヤーの多用により、本来レイヤー化して描画を分離したい要素が意図通りに分離できない事態も発生する可能性がある。

一般的に CSS Animations、CSS Transitions などを指定することによりレイヤー化することができる。他にも translate3d(0,0,0)などの指定により強制的にレイヤー化するハック的な手法もあるため、対象となる端末上で事前に検証しておくことを推奨する。