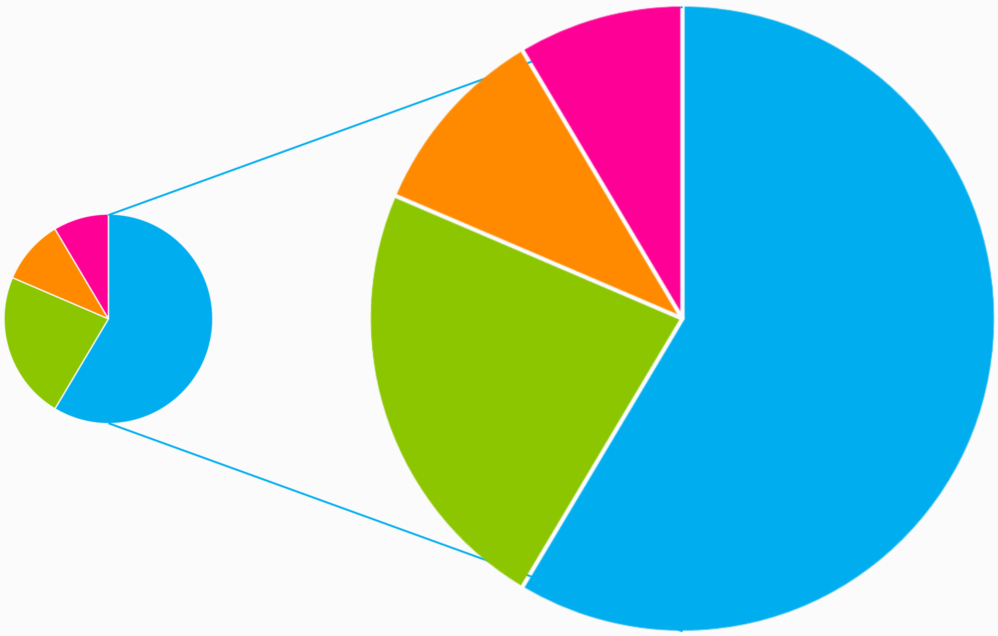
W3C/慶應 デジタルサイネージとHTML5 セミナー

サイネージのためのWebプロファイル

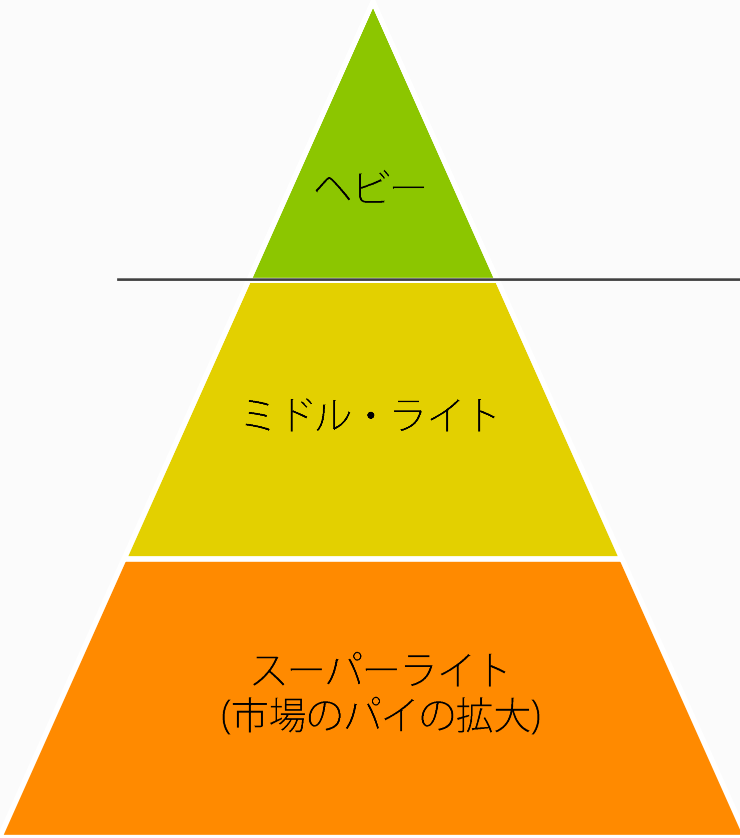
2015年3月26日

株式会社ニューフォリア
取締役 最高技術責任者
羽田野 太巳 (はたの ふとみ)
@futomi futomi.hatano
http://www.newphoria.co.jp/

サイネージ市場拡大を目指して



WBSがターゲットとする市場



屋外
公共性高
巨大なパネル
広告メディア
24-7-365の堅牢なシステム

屋内
公共性中
TVサイズのパネル
情報発信メディア
営業時間のみの運用

屋内
公共性低
タブレットサイズのパネル
情報発信メディア
営業時間のみの運用

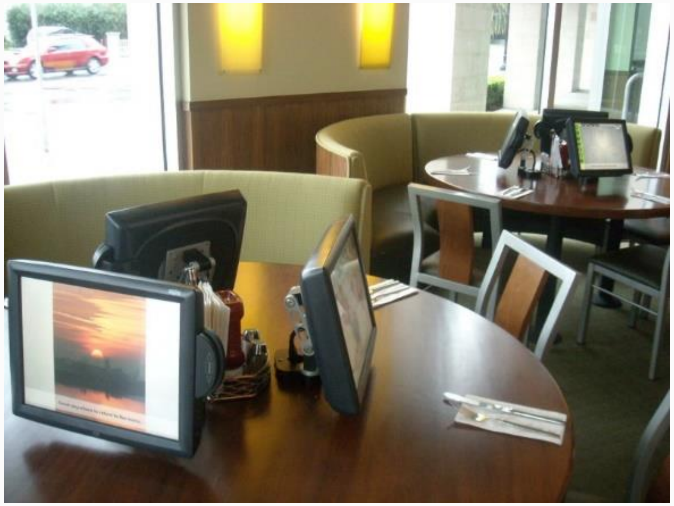
ライトサイネージ市場

- 屋内向けの一般TVサイズのパブリックサイネージ
- オフィス/イベント会場/スーパーマーケット/モール

スーパーライトサイネージ市場

- 屋内向けのタブレットサイズのパーソナルサイネージ

床屋/美容院
(QBハウス)



飲食店

トイレ

WBS普及の鍵

- 端末の低価格化
  - コンシューマー製品をできる限り流用できるソリューション提供
- コンテンツのMulti Use
  - コンテンツの流用が可能なウェブによるアプリ基盤の共通化
  - 標準化技術(HTML5など)とOSSの活用
- Out-of-the-box
  - 導入までの手間を徹底的に省く
  - パッケージ化による選択式サービスの提供
- 新たな価値創出
  - 外部デバイスとの連携など

ウェブ化のメリット(1)

- 標準化技術
  - ブラウザーランタイムがあれば動作可能
  - コンテンツ制作の学習コストの解決
- 各種メディアに対応
  - 静止画、ビデオ、オーディオ、音声合成と生成など
- 表現力
  - ウェブサイトやウェブアプリの表現力をそのままサイネージに

ウェブ化のメリット(2)

- ビデオ依存からの脱却
  - アニメーションならビデオにする必要がない
  - データ量を小さくできる

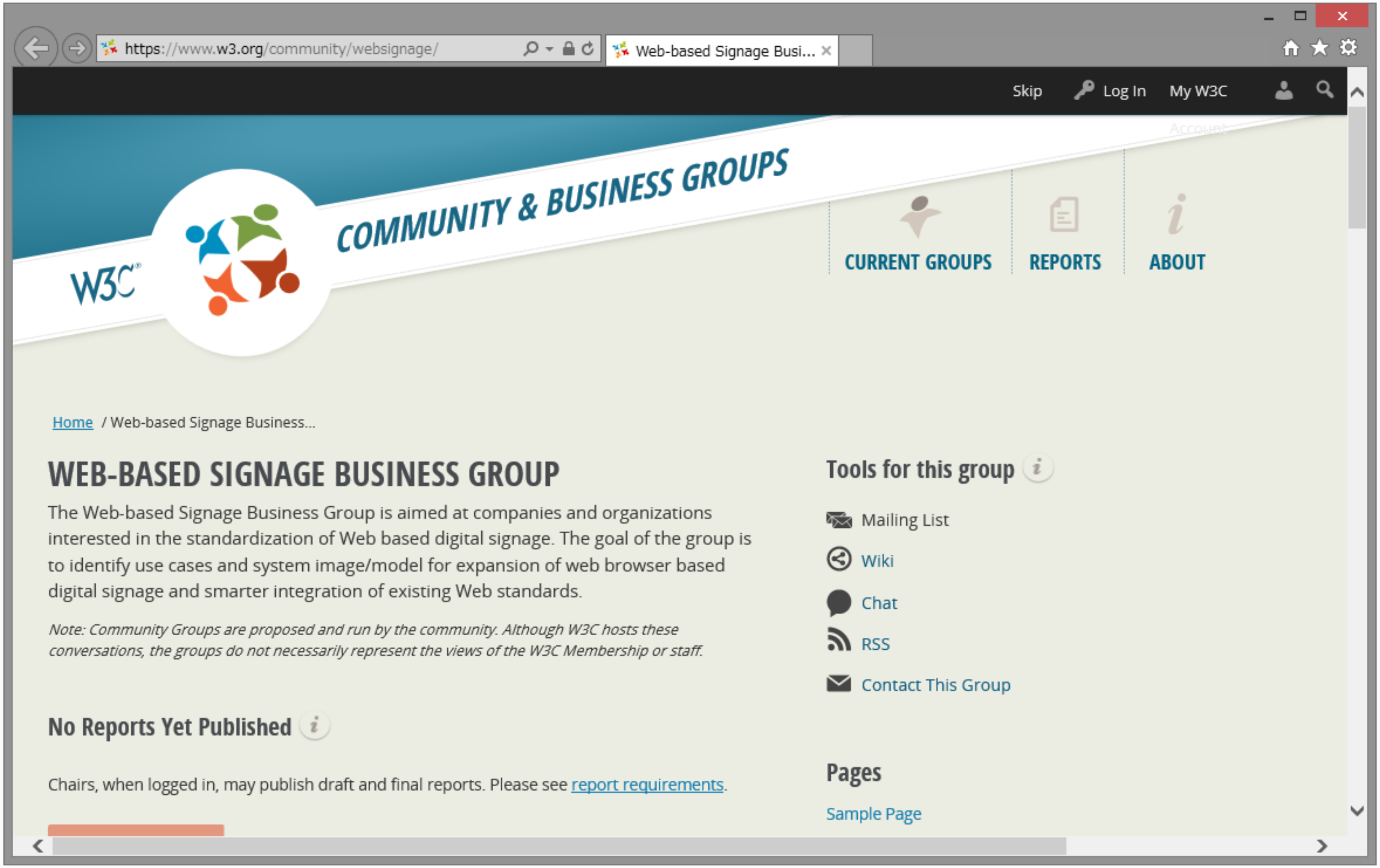
- パネル/STBのマルチファンクション化
  - アプリ基盤をウェブにすることでアプリケーションの自由度が増し、ハードの多目的化・多用途化が実現される

ウェブ化の課題

- ノウハウ
  - サイネージにウェブ技術を使った事例が少ない
  - どうやってサイネージをウェブで実現するのか技術的課題が見えていない
- 組み込み機器
  - テレビ機器などの組み込み機器での制約
  - 機能面とパフォーマンス面での課題

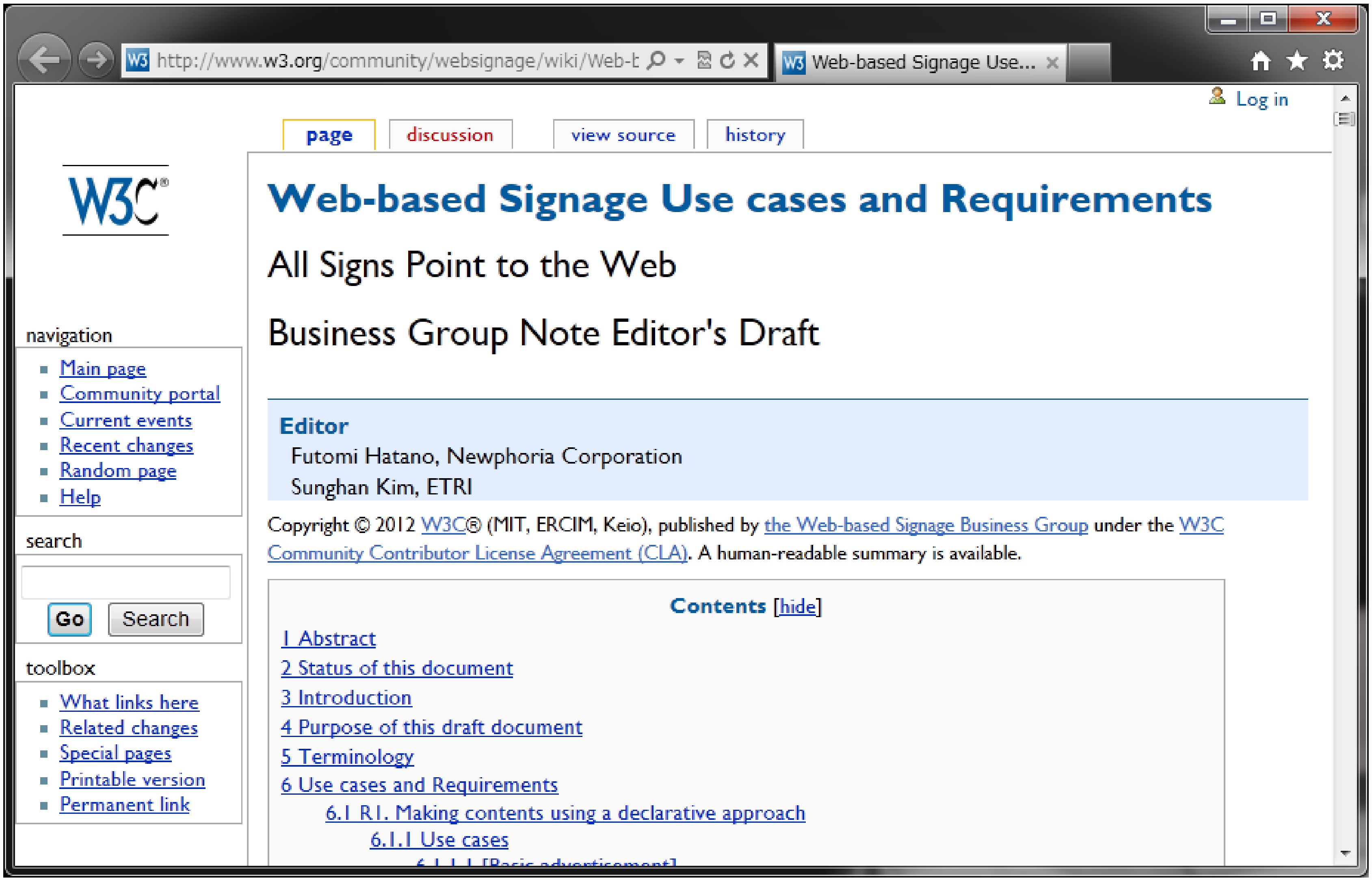
Web-based Signage BG

Web-based Signage BG



http://www.w3.org/community/websignage/

Use cases and Requirements



http://www.w3.org/community/websignage/wiki/Web-b

Web-based Signage Use...

Log in

page | discussion | view source | history

Web-based Signage Use cases and Requirements

All Signs Point to the Web

Business Group Note Editor's Draft

Editor

Futomi Hatano, Newphoria Corporation

Sunghan Kim, ETRI

Contents [hide]

navigation

- Main page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go | Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

http://www.w3.org/community/websignage/wiki/Web-based_Signage_Use_cases_and_Requirements

Use cases and Requirements

- 検討議論の結果…
  - ラフすぎる要件
  - やや未来すぎの話も
  - 実践的ではない

- より具体的で詳細で実践的な要件が必要
- プロファイル作成に着手

プロファイル作成の目的

- 普及
  - 業界に対してウェブ技術応用の啓蒙と普及
- 共有
  - 業界に対してウェブ技術を使ったサイネージの技術知識の共有
- 発見
  - サイネージにウェブ技術を応用する際に不足しているAPIの発見とW3C内での提案活動

プロファイルの用途

- 端末製品諸元表
  - Web-based Signage端末の諸元表などへの活用
- RFP (Request For Proposal)
  - サイネージオペレータがシステムベンダーに発注する際のRFPの記述への応用
- など

Architecture and Requirements
for Web-based Signage

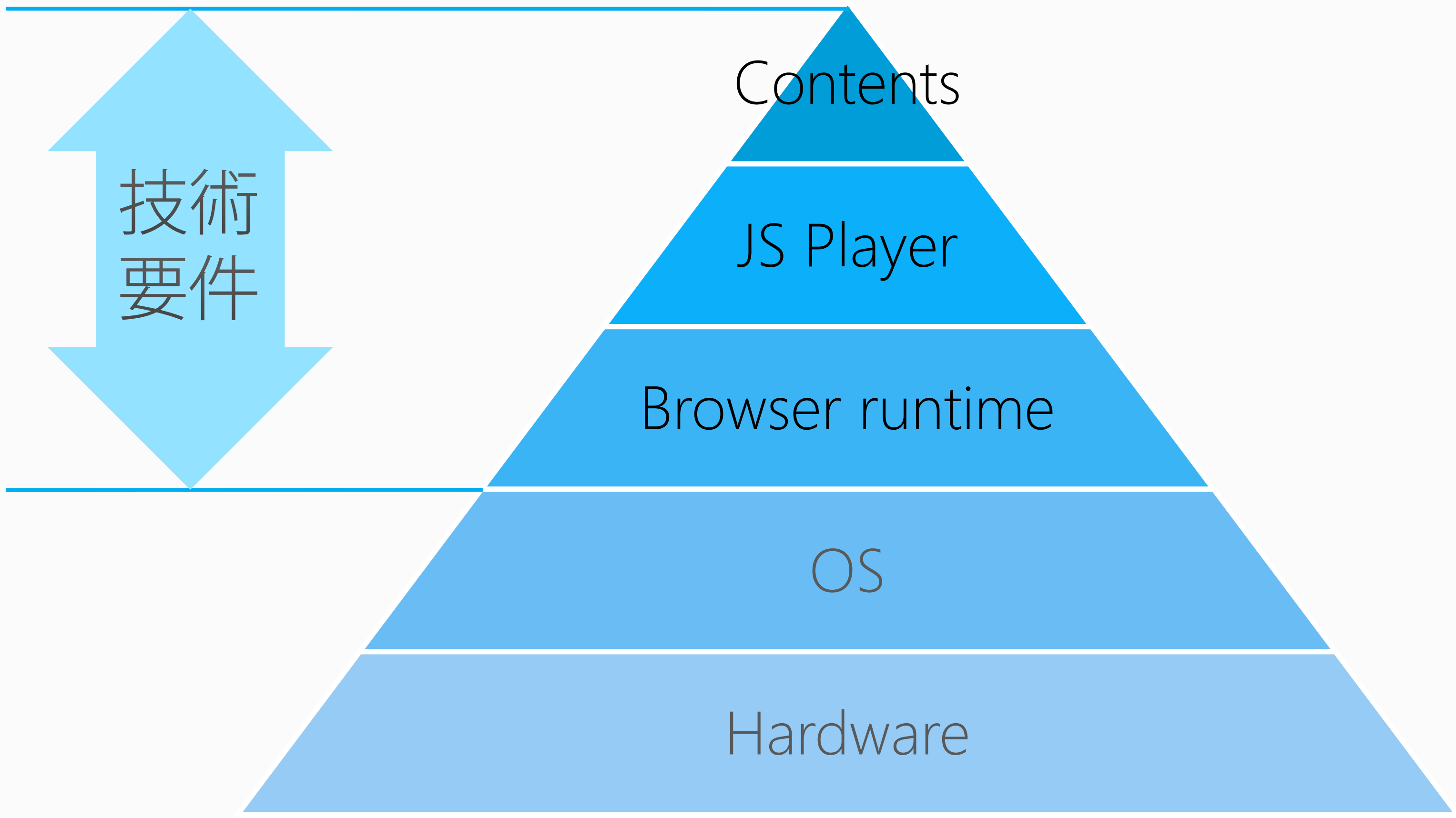
- Core Profile
  サイネージとしての最小限の要件定義
  マルチリージョン、遷移効果、ティッカーなど
- Basic Media Profile
  ビデオとオーディオ再生の要件定義
- Storage Profile
  コンテンツの保存と読み出しにおける要件定義

http://www.w3.org/community/websignage/wiki/Main_Page#Architecture_and_Requirements_for_Web-based_Signage_Player

プロファイル分離の理由

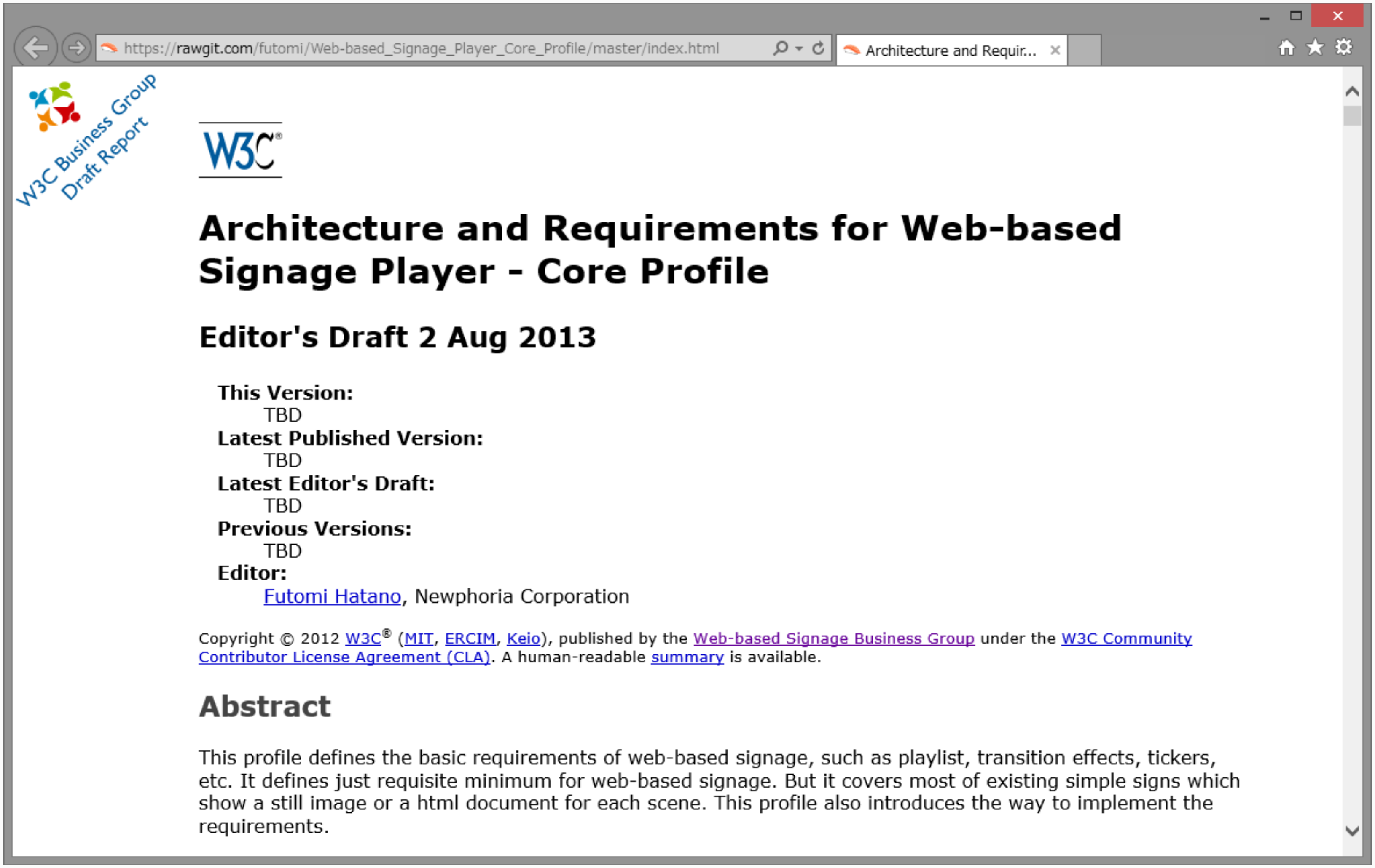
- すべての要件を必要とするシーンは少ない
  - 特にライトサイネージにフォーカスするWBSでは必要とされる機能はシーンによって限定的
- プロファイルの活用のしやすさ
  - すべての要件を満たすハードは高価にならざるをえないため、WBSが目指す方向性に逆行
  - 特定のプロファイルに限定対応したWBS対応ハードの普及を後押し

プロファイルの対象



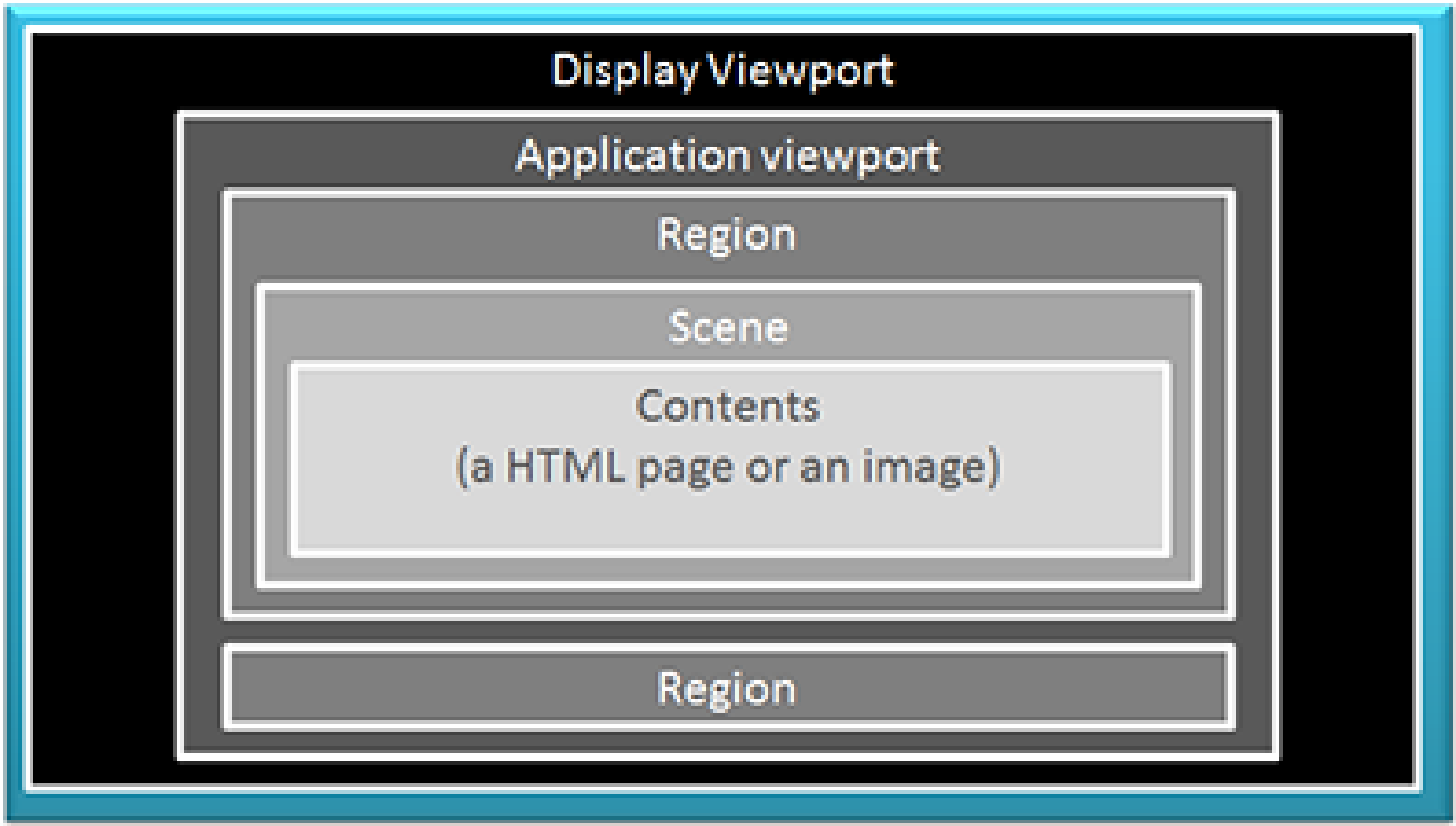
Core Profile

Core Profile



https://rawgit.com/futomi/Web-based_Signage_Player_Core_Profile/master/index.html

ビューポートの概念


Display Viewport
Application viewport
Region
Scene
Contents
(a HTML page or an image)
Region

ビューポートの属性

fit = meet, horizontal align = left

fit = meet, horizontal align = center

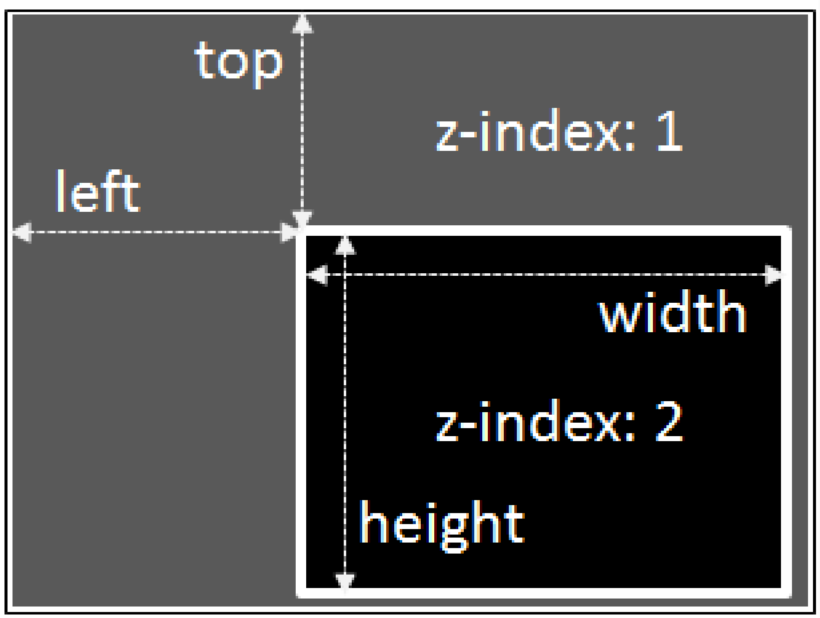
fit = meet, horizontal align = right

fit = fill, vertical align = top

fit = fill, vertical align = middle

fit = fill, vertical align = bottom

リージョン



シーンのコンテンツタイプ

- 画像
  - img要素/object要素
  - PNG, GIF, JPEG
  - DataURL (RFC2397)
- HTMLドキュメント
  - JS Playerはiframe要素でサンドボックス化
  - JS Playerの挙動や表示に影響を与えず、シーン独立で完結させるため

iframeシーンのイベント通知

- シーンの開始、中断、再開、終了
  - JS Playerの親ドキュメントとiframe内のシーンHTMLドキュメントとの間にイベント通知機構が必要
- DOM4の独自イベントとWeb Messaging
  - サンプルコードを提示
  - どちらを使っても良いことに
  - メリットとデメリットを提示

プレイリスト

- Simple loop
  - A -> B -> C -> D -> E -> A -> B -> C -> D -> E -> A -> ...
- Individual loop
  - A -> B -> B-> C -> D -> E
- Nested loop (Optional)
  - A -> B -> C -> B -> C -> D -> E.

スケジューリング

- Scheduling types
  - One shot (Optional)
  - Daily shot
  - Hourly shot
  - Minutely shot (Optional)
- Breaking in the playlist
  - Hard break/ Soft break (Optional)
- Resuming the playlist
  - Strict resume (Optional) / Lax resume

画面遷移効果

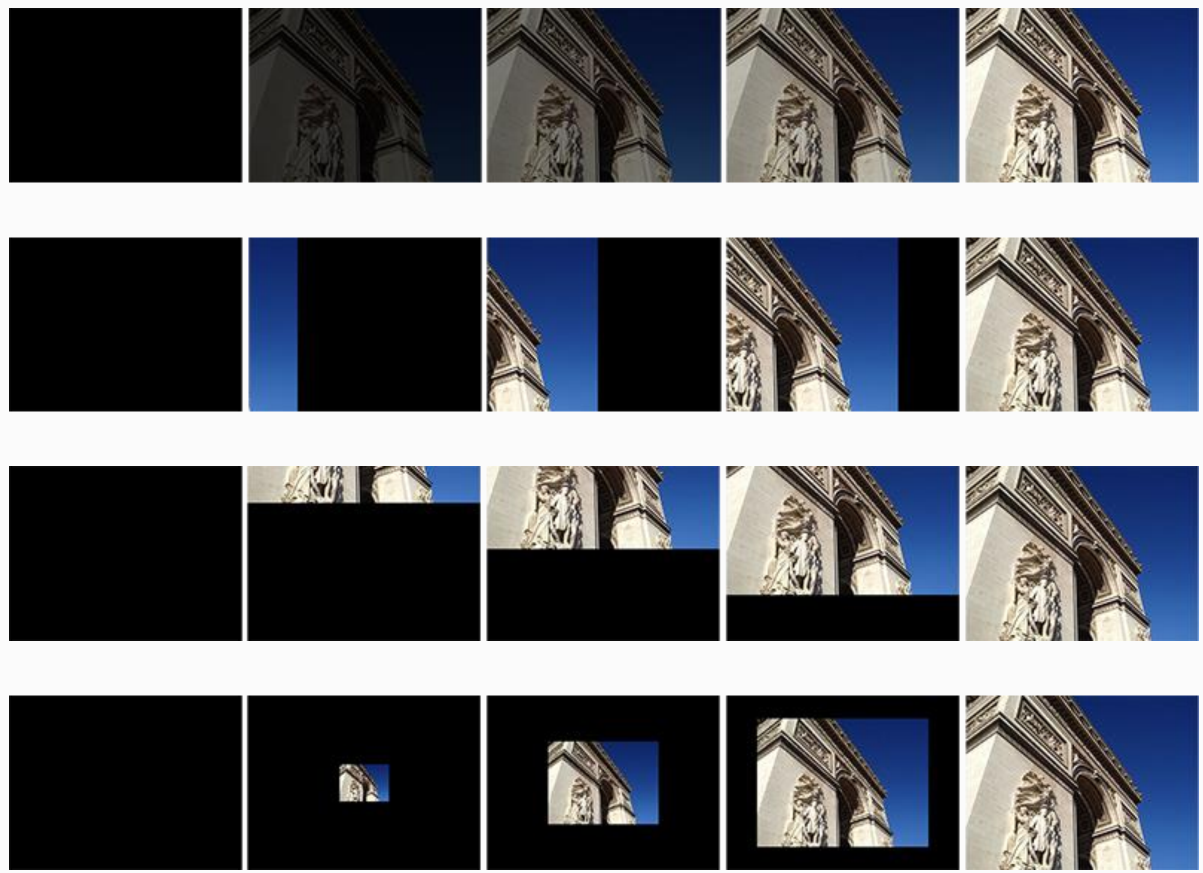
CSS Transitions, CSS Transforms, CSS Color Module

Fade

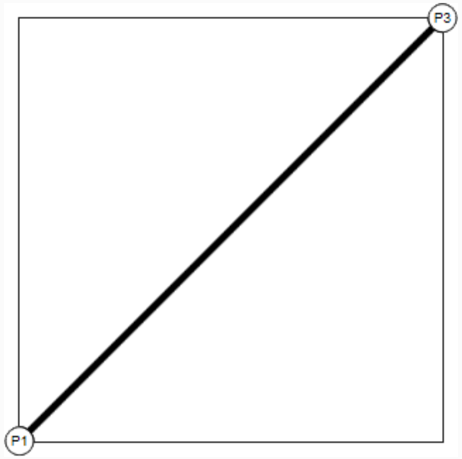
Slide to right

Slide to bottom

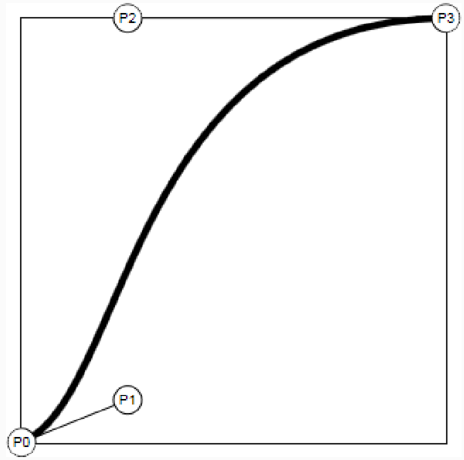
Scale



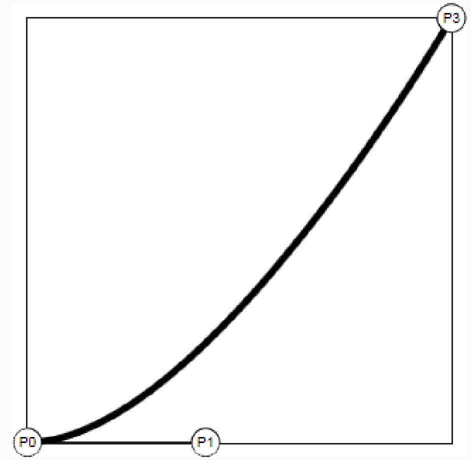
イージング



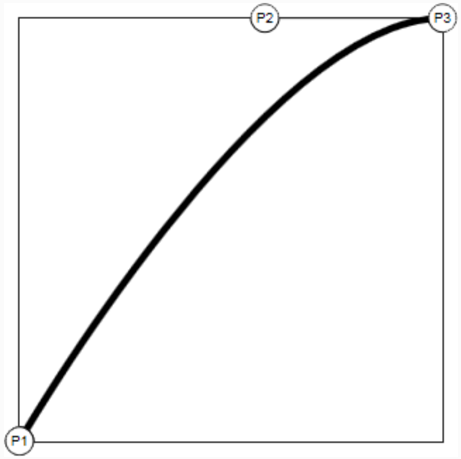
linear



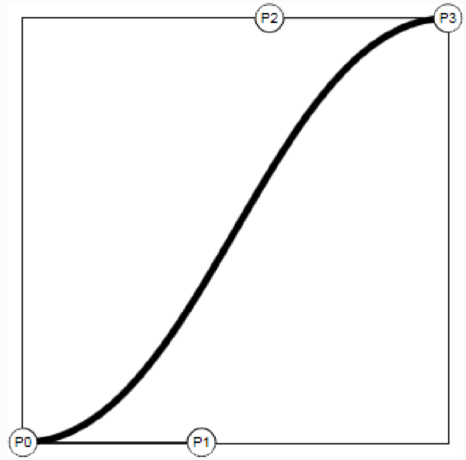
ease



ease-in

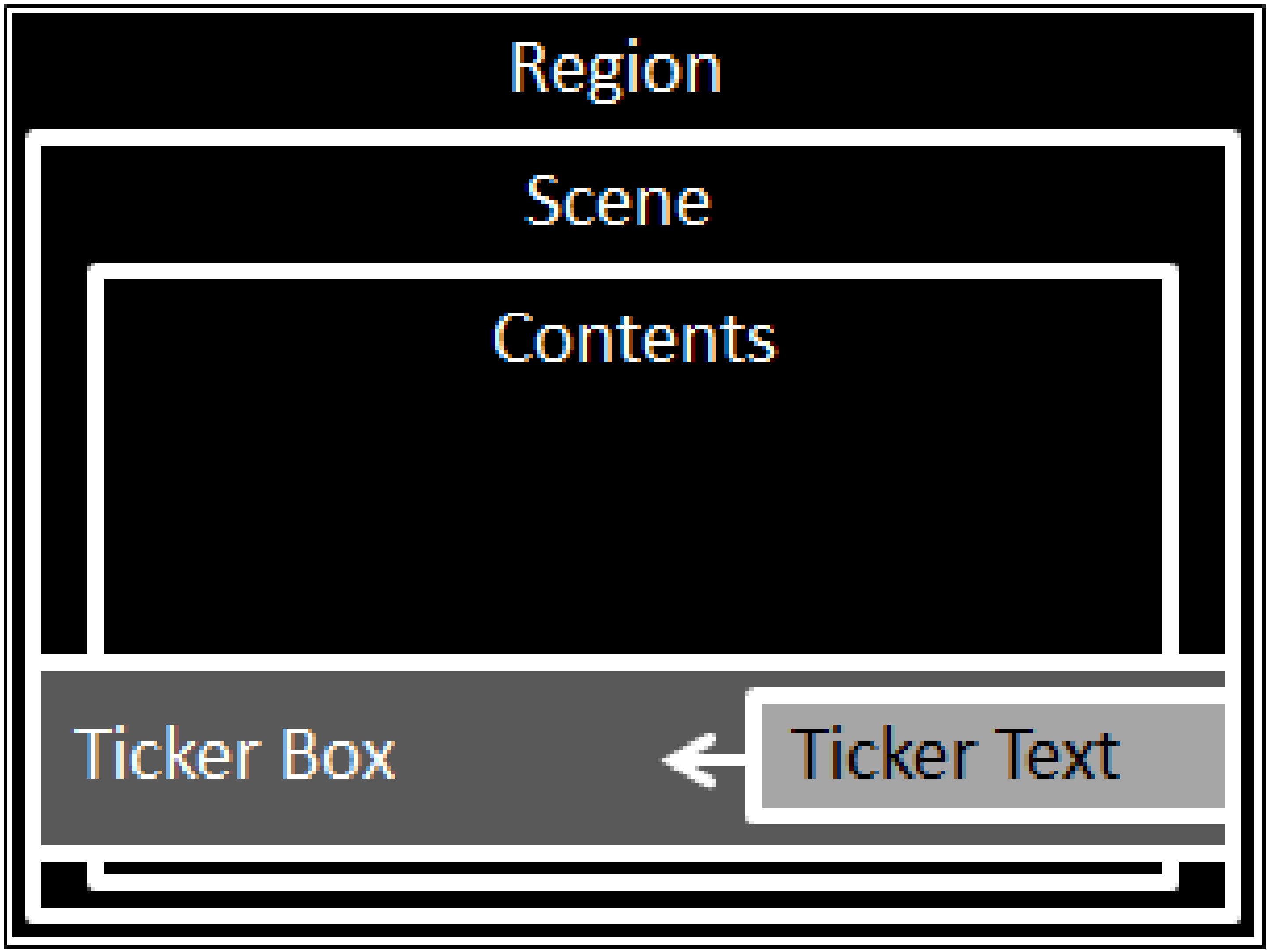


ease-out



ease-in-out

ティッカー



ティッカーテキストの属性

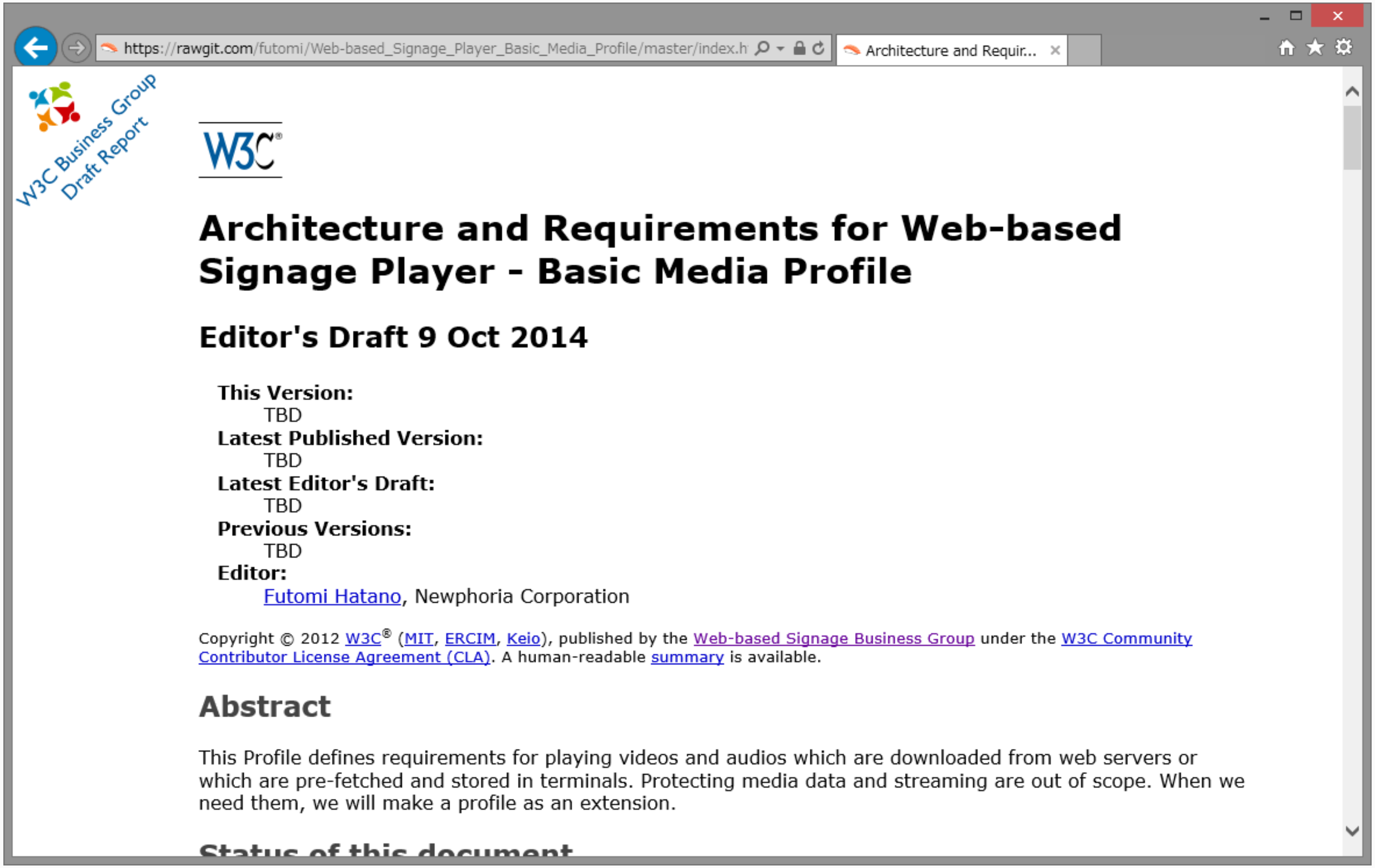
- テキストタイプ
  - フォントと画像テキスト
- 移動速度
  - 秒ごとのピクセル数
- 移動方向
  - LTR (英語や日本語など) とRTL (アラビア語など)

こんにちわ。こんばん

مرحبا. مساء الخي

Basic Media Profile

Basic Media Profile



Architecture and Requirements for Web-based Signage Player - Basic Media Profile

Editor's Draft 9 Oct 2014

This Version:
TBD
Latest Published Version:
TBD
Latest Editor's Draft:
TBD
Previous Versions:
TBD
Editor:
Futomi Hatano, Newphoria Corporation

Abstract

This Profile defines requirements for playing videos and audios which are downloaded from web servers or which are pre-fetched and stored in terminals. Protecting media data and streaming are out of scope. When we need them, we will make a profile as an extension.

Status of this document

https://rawgit.com/futomi/Web-based_Signage_Player_Basic_Media_Profile/master/index.html

メディアタイプ

MIME Type	Video Codec	Audio Codec	Container
video/mp4	H.264	AAC	MP4
video/webm	VP8	Vorbis	WebM

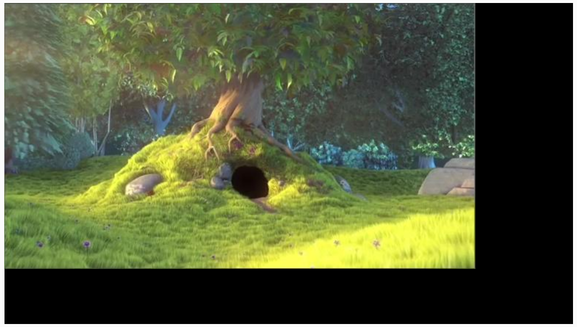
MIME Type	Audio Codec	Container
audio/mp3	MP3	MP3
audio/mp4	AAC	M4A
audio/webm	Vorbis	WebM
audio/ogg	Vorbis	Ogg

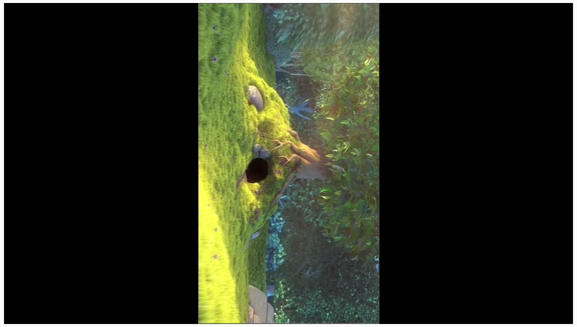
自動再生

- JavaScriptから再生の開始
  - video_el.play()
- モバイル向けブラウザーでは自動再生が禁止
  - データプランによっては従量課金のため、ユーザーの意図しない課金を発生させないため
  - 再生医は画面タッチなどのユーザーアクションが必要
- サイネージでは自動再生の許可は必須
  - モバイル向けOS/ブラウザーを採用するハードウェアはブラウザーランタイムの改造が必要

ビデオの変形

- video要素にCSS Transforms
  - rotate(), scale(), translate()
  - 変形を適用したまま再生できること
  - ディスプレイ縦置きやマルチリージョンに





テストスイート

9 Runtime conformance test suites

The conformant runtime must run the test suites appropriately without any JavaScript error:

Video codec, preload, autoplay test

The conformant runtime must run this test appropriately.

This test checks which video codecs the runtime supports, whether the runtime preloads media data enough to start to play before playing a video, and whether the runtime can start to play a video without any user actions.

The runtime must pass all tests at least for one codec.

Audio codec, preload, autoplay test

The conformant runtime must run this test appropriately.

This test checks which audio codecs the runtime supports, whether the runtime preloads media data enough to start to play before playing an audio, and whether the runtime can start to play an audio without any user actions.

The runtime must pass all tests at least for one codec.

Video scaling test

The conformant runtime must run this test appropriately.

This test checks whether the video is scaled appropriately setting the `width` and `height` attribute of the `HTMLVideoElement` object.

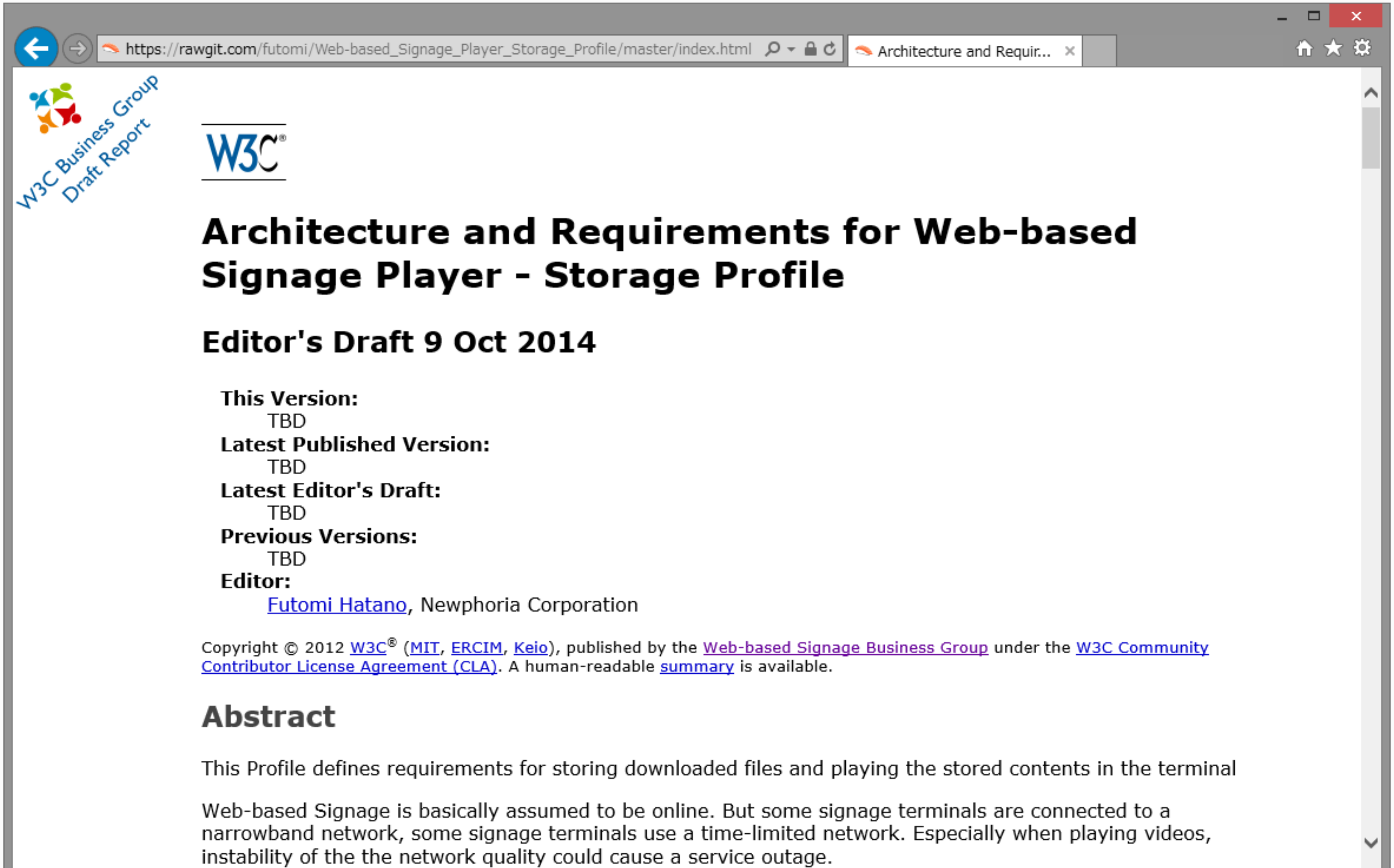
Video transforming test

The conformant runtime must run this test appropriately.

This test checks whether `translate()`, `scale()`, `rotate()`, and the combination specified in *CSS Transforms* affect a video appropriately.

Storage Profile

Storage Profile



https://rawgit.com/futomi/Web-based_Signage_Player_Storage_Profile/master/index.html

Architecture and Requir...

W3C Business Group Draft Report

W3C®

Architecture and Requirements for Web-based Signage Player - Storage Profile

Editor's Draft 9 Oct 2014

This Version:
TBD
Latest Published Version:
TBD
Latest Editor's Draft:
TBD
Previous Versions:
TBD
Editor:
Futomi Hatano, Newphoria Corporation

Abstract

This Profile defines requirements for storing downloaded files and playing the stored contents in the terminal

Web-based Signage is basically assumed to be online. But some signage terminals are connected to a narrowband network, some signage terminals use a time-limited network. Especially when playing videos, instability of the the network quality could cause a service outage.

https://rawgit.com/futomi/Web-based_Signage_Player_Storage_Profile/master/index.html

データのロードと保存

3 Storing contents

The application gets a content file from the CDN using *XMLHttpRequest*. Setting the `responseType` attribute of the `XMLHttpRequest` object to "`blob`", the application gets the file as a Blob object.

XMLHttpRequest code example

```
loadVideoFile();

// Download a video file as a blob object using XHR
function loadVideoFile() {
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "v.mp4");
    xhr.responseType = "blob";
    xhr.onload = function() {
        if(xhr.status === 200) {
            var blob = xhr.response;
            // Save the blob to the storage
            if(window.indexedDB) {
                saveBlobIdb(blob);
            } else if(window.webkitRequestFileSystem) {
                saveBlobFsys(blob);
            }
        }
    };
    xhr.send();
}
```

The application can use *Indexed Database API* or *File API: Directories and System* to store files. These APIs allow scripts to store Blob objects directly.

The runtime must support *XMLHttpRequest*. The runtime must support *Indexed Database API* or *File API: Directories and System*. Besides the runtime must support storing Blob objects directly, which is specified in *File API*. [XHR] [IDB] [FILEDIR] [FILE]

蓄積コンテンツの再生

4 Playing the stored contents

4.1. Indexed Database API

If the application uses *Indexed Database API* to store files, it gets the file as a Blob object. The application creates a blob URL from the Blob object using `URL.createObjectURL()` method, which is specified in *File API*. Then The application creates a appropriate HTML element (e.g. `video` element if a video file, `audio` element if a audio file, `img` element if a image file) and sets the `src` attribute to the blob URL.

Indexed Database API code example

```
// Get the blob from the DB
function getBlobIdb(file_name) {
    var db_name = "mydb";
    var store_name = "mystore";
    // Open the DB
    var idb = window.indexedDB;
    var openreq = idb.open(db_name, 1);
    // Get the blob object
    openreq.onsuccess = function(event) {
        var db = event.target.result;
        var transaction = db.transaction([store_name], "readonly");
        var store = transaction.objectStore(store_name);
        var getreq = store.get(file_name);
        // Create a blob URL and play the video
        getreq.onsuccess = function(event) {
            var rec = event.target.result;
            var blob_url = window.URL.createObjectURL(rec["data"]);
            playVideoIdb(blob_url);
        };
    };
}
```

CORS

5 Cross-Origin Resource Sharing

In practice, A web-based signage system consists of not only players but also content delivery servers and content management systems. In some cases, script files associated with the application could be hosted in other servers. That is, the origins of contents and application are not necessarily same.

XMLHttpRequest normally does not allow applications (scripts) to fetch files whose origin is different from the application. To do so, the runtime and content delivery servers must support *Cross-Origin Resource Sharing*. The conformant runtime must support at least Origin Request Header and Access-Control-Allow-Origin specified in *Cross-Origin Resource Sharing*. [CORS]

ストレージ容量管理

6 Quota management

When the application stores resource files (e.g. video file, image files), the application needs to know the total size, used size, and free size of the storage.

Quota Management API meets this requirements. *Quota Management API* allows scripts to get the total size and the used size of the storage. The free size is available subtracting the free size from the total size.

If runtime supports *Quota Management API*, the application can get the information about the quota using the `queryInfo()` method of the `StorageQuota` interface.

Quota Management API code example

```
navigator.storageQuota.queryInfo("temporary").then(
    function(info) {
        console.log("Total: " + info.quota + " bytes");
        console.log("Used : " + info.usage + " bytes");
        console.log("Free : " + (info.quota - info.usage) + " bytes");
    }
);
```

The runtime must support *Quota Management API*. [QUOTA]

テストスイート

7 Runtime conformance test suites

The conformant runtime must run the test suites appropriately without any JavaScript error:

Fetching and storering test

All of the conformant runtime must run this test appropriately.

This test downloads a video file as a blob object using *XMLHttpRequest*. Then the file is stored using *Indexed Database API* if supported. If not, this test tries to use *File API: Directories and System*.

If the file is stored successfully, this test tries to get the file as a blob object from the storage, to create a `video` element, then to play the video.

CORS test

All of the conformant runtime must run this test appropriately.

This test downloads an image file as a blob object using *XMLHttpRequest* from the site whose origin is different from this test script.

If the image file is downloaded successfully, it will be shown in this test page.

Quota Management test

All of the conformant runtime must run this test appropriately.

This test tries to call the `queryInfo()` method of the `StorageQuota` interface specified in *Quota Management API*. If succeeded, the total size, the free size, and used size will be shown.

ご清聴ありがとうございました

株式会社ニューフォリア
取締役 最高技術責任者
羽田野 太巳 (はたの ふとみ)

@futomi　futomi.hatano

http://www.newphoria.co.jp/