モダンブラウザにおいて採取可能な端末情報の調査: ２０１４年版

高須 航† 磯 侑斗† 桐生直輝† 塚本耕司† 武居直樹† 山田智隆† 齋藤孝道‡

†明治大学大学院，‡明治大学
214-8571 神奈川県川崎市多摩区東三田 1-1-1
{ce46021, ce36004, ce36022, ce36032, ce46022, ce46035}@meiji.ac.jp
saito@cs.meiji.ac.jp

あらまし 近年, WebページにおけるJavaScriptやHTML5の利用が増えてきた. その一方で, これら技術は，画面サイズやバッテリー状況など利用者の端末情報を採取するための手段として用いることが可能であり，利用者の意思とは別にWebサーバはそれら情報を採取することができる状況にある．また，我々の研究により，ハードディスクの空き容量やCPUのコア数といった従来のWeb技術では採取し得なかった情報も採取できることが判明している．本論文では, JavaScript及びHTML5を用いて利用者のブラウザから採取できる様々な情報について，具体的な採取方法と合わせて報告する.

A Survey of Hardware Characteristics in Modern Browsers

2014 Edition

Ko TAKASU† Yuto ISO† Naoki KIRYU† Koji TSUKAMOTO† Naoki TAKEI†
Tomotaka YAMADA† Takamichi SAITO‡

†Graduate School of Meiji University, ‡Meiji University
1-1-1, Higashi-Mita, Tamaku, Kawasakishi, Kanagawa 214-8571, JAPAN
isd@isc.meiji.ac.jp, saito@cs.meiji.ac.jp

Abstract Recently, JavaScript with HTML5 is getting popular to be utilized in building Web site. As a side effect, those technologies can be used to collect information, so called fingerprint, about hardware of a device such as battery status or screen size. Web server can collect these kinds of information regardless of the user's agreement. Moreover, our studies show that we can also collect information that could not be obtained with existing fingerprinting, e.g., the number of CPU cores, hard disk space and so on. In this paper, we report that we can collect various information using with HTML5 APIs.

1 はじめに

近年の多くの Web ページで JavaScript が利用されている．W3Techs[7]によると，Alexa[8]の提供する，Web サイトの人気ランキング上位 1 千万サイトのうち，2014 年現在における JavaScript の利用率は 88.1%まで達していることが分かる．また，2014 年までの正式勧告を目指して仕様策定が行われている HTML5 では，様々な API の仕様策定

も行われており，今日のモダンブラウザではより高度な Web アプリケーションを実現できるようになった．

これらの Web 技術は画面サイズやバッテリーの状況など利用者の端末に関する様々な情報を採取するための手段として用いることが可能であるが，副作用として，利用者の意思とは別に，Web サーバが様々な情報を採取することができる状況とも言える．2014 年現在，HTML5 などを用いて，より多くの端末情報を採取することが可能である．このような状況は，Web サーバにおいて Web ブラウザもしくはそのデバイスを特定する行為，すなわち，Web Browser Fingerprinting に繋がるリスクであると言える．ここで，Web Browser Fingerprinting とは，Web サーバへアクセスした際に，Web サーバが Web ブラウザから情報（特に，端末や Web ブラウザの利用者の特定に繋がるような）を採取する行為をいう．また，我々の研究において，HTML5 API を利用することでハードディスクの空き容量[1]や CPU のコア数[2]といった従来の Web 技術では採取し得なかった情報も採取可能であることが判明している[4]．

本論文では，JavaScript 及び HTML5 を用いて利用者のブラウザから採取できる様々な情報について，具体的な採取方法と併せて報告する．ここで，HTML5 API とは HTML5 の仕様策定に伴い新たに策定された JavaScript から操作する API の総称である．HTML5 API という言葉に明確な定義はなく，W3C や WHATWG の他，IETF や Khronos などにより策定された API も含めて HTML5 API と呼称されることもある．多くの API は勧告に至っていないが，いくつかのブラウザベンダによって先行実装されており，ベンダプレフィックスをプロパティの接頭辞として付加し，利用することができる．

2 端末情報の採取方法

本節では，JavaScript と HTML5 API から採取可能な端末情報と，それぞれの採取方法について説明する．端末情報を採取する Web サーバは，利用者の Web ブラウザ上で動作させた結果を自身に送信させることで，その実行結果を採取することができる．端末情報は，[4]に倣って，Web ブラウザに関する情報，ネットワークに関する情報，ハードウェアに関する情報という 3 つの種類に分類した．なお，本論文では，利用者の許可が必要となる API は扱わないこととする．

2.1 Web ブラウザに関する情報

2.1.1 ユーザエージェント

HTTP リクエストヘッダと JavaScript の navigator.userAgent から採取することができる．採取した文字列から，OS の種類，Web ブラウザの種類，ブラウザのバージョン，言語設定を読み取ることができる．

2.1.2 HTTP クッキーが利用可能か否か

navigator.cookieEnabled から採取できる．クッキーが利用可能であれば true，利用不可能であれば false が返る．

2.1.3 WebStorage が利用可能か否か

Web ブラウザには，利用者のローカル環境にデータを保存する仕組みとして WebStorage[9]が実装されており，Web ブラウザのタブが閉じられるまで保存される SessionStorage と，利用者が明示的に削除しない限り永続的に保存される LocalStorage がある．window.sessionStorage と window.localStorage からそれぞれ採取できる．利用可能であれば何らかのオブジェクトが返り，利用不可能であれば何も返らない．

2.1.4　プラグイン

Plugin オブジェクトから，ブラウザにインストールされているプラグインに関する情報を採取することができる．図 1 にプラグイン情報を採取するコードを記載する．

```
1  var pluginlist = '', i, j;
2  var p = navigator.plugins;
3  for (i = 0; i < p.length; i++) {
4    pluginlist += p[i].name +
5    p[i].description + p[i].filename +
6    p[i].version;
7    for (j = 0; j < p[i].length; j++) {
8      pluginlist += p[i][j].description +
9      p[i][j].type + p[i][j].suffixes;
10   }
11 }
```

図 1．プラグイン情報を採取するコード

p.length はブラウザにインストールされているプラグインの個数を表しており，Plugin オブジェクトの name, description, filename, version プロパティから，プラグインの名前，説明，ファイル名，バージョンを採取することができる．また，p[i].length は各プラグインが持つ MIME タイプの個数を表しており，MIME タイプオブジェクトの description, type，suffixes プロパティから，MIME タイプの説明，MIME タイプ名，拡張子を採取することができる．

2.2　ネットワークに関する情報

2.2.1　プライベート IP アドレス

WebRTC を用いることで採取することができる．図 2 にプライベート IP アドレスを採取するコードを記載する（図 2 参照）．

プライベート IP アドレスとは，LAN 内でのみ使用される IP アドレスである．インターネットにアクセスする際はグローバル IP アドレスに変換されるため，一般的に通信相手が知り得ることはない．

onicecandidate や createOffer の第 1 引数はコールバック関数であり，引数として Session Description Protocol（SDP）[11]形式のネットワーク情報が渡される．SDP からクライアント端末のプライベート IP アドレスを得ることができる．

```
1  var RTCPeerConnection =
2    window.RTCPeerConnection ||
3    window.webkitRTCPeerConnection ||
4    window.mozRTCPeerConnection;
5  if (RTCPeerConnection) {
6    var rtc = new
7          RTCPeerConnection({iceServers:[]});
8    if (window.mozRTCPeerConnection)
9      rtc.createDataChannel('',
10                              {reliable:false});
11   rtc.onicecandidate = function (evt) {
12     if (evt.candidate)
13       console.log(evt.candidate.candidate);
14   };
15   rtc.createOffer(function (offerDesc) {
16     console.log(offerDesc.sdp);
17     rtc.setLocalDescription(offerDesc);
18   }, function (e) {
19     console.warn("offer failed", e); });
20 }
```

図 2．プライベート IP アドレスを採取するコード

2.3　ハードウェアに関する情報

2.3.1　画面解像度・色深度

画面解像度は端末の画面の縦サイズ×横サイズで表され，screen.height, screen.width よりそれぞれ採取することができる．色深度とは，1 ピクセルで表現できる色の数を表しており，screen.colorDepth より採取することができる．

2.3.2　ディスプレイの向き

端末のディスプレイが既定の向きか，時計回りに 90°回転しているか，反時計回りに 90°回転しているか，逆さまかという情報を採取することができる．Internet Exolorer（IE）と Firefox が対応しており，それぞれ screen.msOrientation, screen.mozOrientation から採取することができる．

2.3.3　タッチパネルの有無

document オブジェクトに Touch Events の ontouchstart イベントが含まれているか否かで，タッチパネルの有無を採取することができる．別の手法として，タッチパネルに同時にタッチすることができる最大の数を返す navigator.maxTouchPoints メソッドから採取することもできる．

2.3.4　フォント

端末にインストールされたフォント一覧の採取には Adobe Flash Player などのような外部のプラグインが必要であるが, IE ではバージョン 6 以降から利用できる Dialog Helper Object を用いることで，JavaScript から採取することができる．図 3 にフォントを取得する，IE 向けのコードを記載する．

```
 1  <body>
 2    <object id="dlgHelper"
 3      CLASSID="clsid:3050f819-98b5-11
 4      cf-bb82-00aa00bdce0b" width="0px"
 5      height="0px">
 6    </object>
 7    <script>
 8      var fontslist ='';
 9      for (i = 1; i < dlgHelper.fonts.count;i++) {
10        fontslist += dlgHelper.fonts(i) + ',';
11      }
12    </script>
13  </body>
```

図 3. フォントを採取するコード

図 3 で示した object タグは, Dialog Helper オブジェクトを読み込むために必要なタグである．for 文内の count プロパティはフォントの数を表しており，dlgHelper オブジェクトの fonts プロパティから，端末にインストールされているフォントを採取することができる．

この他に，フォントの種類によって Web ページに描画される文字列の縦幅と横幅の組み合わせが異なることを利用したフォントの識別手法が存在する[14]．CSS（Cascading Style Sheets）の font-family プロパティから，指定したフォントを適用させることが可能であり，JavaScript の fontFamily プロパティから動的に変更することができる．指定したフォントが利用者の端末にインストールされていない場合は, Web ブラウザで設定されているデフォルトのフォントが適用される．また，描画する文字列の縦幅と横幅は，offsetHeight プロパティと offsetWidth プロパティから取得することができる．以上を用いて，特定の文字列を用意しておき，デフォルトのフォントで描画される文字列の縦幅と横幅, font-family プロパティから適用されたフォントの文字列の縦幅と横幅をそれぞれ比較することで, フォントの識別が可能となる．

また，CSS のみで採取する方法も提案されている[4].

2.3.5　デバイスピクセル比

デバイスピクセル比とは，CSS で定義されている論理的なピクセル（CSS ピクセル）に対する，端末自体の物理的なピクセル（デバイスピクセル）の比率である．これは，window.devicePixelRatio から採取することができる．iOS では，非 Retina ディスプレイは 1，Retina ディスプレイは 2 となっている．Android 端末では機種ごとに 1.5，2，3 など様々な値をとる．また，PC の Web ブラウザでは Web ページを拡大/縮小することでこの値が変化する．一部のブラウザでは，拡大/縮小の比率がページごとに保存されるため以降のアクセスでもデバイスピクセル比は変化しない．

2.3.6　SSE2 が利用可能か否か

文献[5]より，SSE2 が利用可能か否かを識別する手法が示されている．利用可能である場合，Pentium4 以降の CPU であり，利用不可能な場合 Pentium4 より前の CPU であることが分かる．

2.3.7　CPU のコア数

CPU の処理性能の差を利用し, 利用者の端末の CPU コア数及び Hyper Threading Technology（以下，HTT と記す）の有効/無

効を推定することができる[2][4][5]．複数のプロセスによって CPU に高負荷な処理を実行させ，その処理にかかった時間の差によって推定する．JavaScript でマルチスレッド処理を可能にする Web Workers API を用いることで高負荷な処理を行い，処理時間を測定する．

1 つのスレッドによる処理時間を 1 としたとき，2～16 スレッドによる処理時間の比を求めることで，物理コア数の推定を行うことが可能となる．また，整数演算と浮動小数点数演算を行わせることによって, HTT の有効/無効を推定することが可能となる．CPU の演算ユニット内には整数演算を行うユニットと浮動小数演算を行うユニットが独立して存在する．HTT は，整数演算と浮動小数演算を同時に実行することができるため，有効時と無効時で処理時間に差が生じる．

2.3.8　Canvas Fingerprinting

HTML5 の Canvas 機能を用いて文字列や画像の描画を行い, その結果の違いから Web ブラウザや OS の種類を識別可能であることが，文献[10]により示されている．また，canvas 要素内に 3D 描画を行うことができる WebGL を用いることで，端末の GPU の種類を特定することができる．

2.3.9　バッテリーステータス

Battery Status API を用いる．この API で定義されている window.navigator.battery プロパティにおける, charging, chargingTime, discharging Time，level メソッドを用いることで，バッテリーが充電中か否か，バッテリーの充電完了までの残り時間，バッテリーが利用できる残り時間，バッテリー残量を採取することができる[4].

2.3.10 ハードディスク空き容量

Web ブラウザのオリジン[6]毎に保存可能なデータ容量の上限値をバイト単位で任意に要求することが可能である，Quota Management API を用いる[4]．図 4 にストレージ空き容量を採取するコードを記載する．

```
1  navigator.webkitTemporaryStorage.re
2  questQuota(
3    100000000000000000,
4    function successCallback(bytes) {
5      hdd_free_space = bytes + '0';
6    },
7    function errorCallback() {
8      hdd_free_space = 'failed';
9    }
10 );
```

図 4．ハードディスク空き容量を採取するコード

この API で定義されている navigator.webkitTemporaryStorage.requestQuota メソッドによりデータ容量の上限値を要求すると，要求した上限値分の保存領域が確保できるか否かを確認することができる[4]．確保できる場合，要求した上限値が successCallback 関数の引数に返る．確保できない場合 errorCallback 関数が実行されるが，ハードディスク空き容量を超える大きな値を要求した場合，確保可能な最大の上限値を要求した場合の上限値が返る．この上限値は，ハードディスク空き容量を約 1MB の誤差で推測可能である．また，Chromium 32.0.1700.76 以降のバージョンでは，図 4 より得られる値の 1.5 倍した値が端末のハードディスク空き容量となっている．

2.3.11 ディスプレイのリフレッシュレート

Animation Timing API を用いることで推測が可能である[4]．リフレッシュレートとは，ディスプレイが単位時間あたりに描画される回数を示す．この API で定義されている requestAnimationFrame メソッドは，ディスプレイが描画されるタイミングで実行される．現在の時刻を取得する Date.now()メソッドを利用し，requestAnimationFrame メソッドが実行された後の時刻と，次にそのメソッドが実行された後の時刻を取得する．それらの差が，ディスプレイの描画時間となる．測定した描画間隔は, CPU の負荷によって誤

差が生じるため，繰り返し測定し，その平均値を計算することで精度を高めることができる．CPU の負荷が大きい場合は，実際のディスプレイのリフレッシュレートを推測可能な値を採取することができない可能性がある．

2.3.12 端末のカメラとマイクの個数

Media Stream Track を用いる．図 5 にカメラとマイクの個数を採取するコードを記載する．

```
1  MediaStreamTrack.getSources(function
2  (media_sources) {
3    for (i = 0; i < media_sources.length; i++) {
4      var media_source = media_sources[i];
5      id = media_source.id
6      kind = media_source.kind;
7    }
8  })
```

図 5．端末のカメラとマイクの個数を採取するコード

MediaStreamTrack オブジェクトの getSources プロパティでは，引数にデバイス情報が渡される．for 文以下では，端末に存在するデバイスの種類（カメラもしくはマイク）と，デバイス一つ一つに生成される GUID を採取している．この GUID は Web ブラウザによって生成され，利用者が明示的に削除しない限り，同じ値を保持し続ける．

3 Web ブラウザの採取状況

端末情報の採取に用いる JavaScript と HTML5 API の一部は，特定の Web ブラウザにのみ実装されているものがある．本章では，我々の調査により確認出来た対応 Web ブラウザの種類（IE, Firefox, Chrome, Safari, 及び Opera に限定）と，対応する最も古いバージョンについて，表 1 に示す．ただし，古いバージョンのインストールが困難である Chrome については確認できていない．現時点では対応するものの，対応したバージョンが確認できていない Web ブラウザについては，「対応」と記す．また，表中に「M」の付いたバージョンは Mozilla のサイト[12]，「C」の付いたバージョンは caniuse.com[13]を参考にした．

4 まとめ

本論文では，JavaScript と HTML5 API を用いて利用者の Web ブラウザから採取できる様々な情報について，具体的な採取方法と合わせて報告した．Web ブラウザの種類やバージョンによって採取できる情報と採取できない情報が多々あるが，多くの API は標準化に向けて仕様策定が行われており，今後様々な Web ブラウザに対応していくと考えられる．また，本論文で報告した端末情報は，端末の推定に利用できる可能性が考えられる．

参考文献

[1] 高須航，磯侑斗，桐生直輝，齋藤孝道，2014，HTML5 APIにより取得可能なデバイス情報を利用した端末識別手法の提案と実装，第76回情報処理学会全国大会公演論文集 3-651，3-652

[2] 後藤浩行，齋藤孝道，2013，Web行動追跡のためのハードウェア特徴点の抽出，2013暗号と情報セキュリティシンポジウム概要集 p72

[3] 桐生直輝，後藤浩行，齋藤孝道，2013，CPU拡張命令の対応の有無によるCPUアーキテクチャの推測，第75回情報処理学会全国大会公演論文集 3-613，3-614

[4] 齋藤孝道，磯侑斗，桐生直輝，Web Browser Fingerprinting に関する技術的観点での一考察，2014年暗号と情報セキュリティシンポジウム予稿集 p93

[5] 桐生直輝，磯侑斗，金子洋平，齋藤孝道，2014, Web Workers を用いた演算処理性能の差によるCPUコア数の推定

[6] RFC6454 The Web Origin Concept

[7] http://w3techs.com

[8] http://www.alexa.com

[9] http://dev.w3.org/html5/webstorag

[10] https://cseweb.ucsd.edu/~hovav/dist/canvas.pdf
[11] http://tools.ietf.org/html/rfc4566.html
[12] https://developer.mozilla.org
[13] http://caniuse.com
[14] http://www.lalit.org/lab/javascript-css-font-detect

表 1. 端末情報の採取可能な Web ブラウザの種類及びバージョン

Web ブラウザ / 端末情報	Internet Explorer	Firefox	Chrome	Safari	Opera
ユーザエージェント	対応	対応	対応	対応	対応
Session Storage	7，8[M]	2	5[M]	4	10.5
Local Storage	7，8[M]	3.5	4[M]	4	10.5
プラグイン	対応	対応	対応	対応	対応
プライベート IP	非対応	22	23[C]	非対応	18.0.1284.49
画面解像度，色深度	対応	対応	対応	対応	対応
ディスプレイの向き	11	14.0.1	非対応	非対応	非対応
タッチパネルの有無 (Touch Events)	22[M]	18[M]	非対応	非対応	15[C]
タッチパネルの有無 (navigator.maxTouch Points)	10[M]	29[M]	35[M]	非対応	対応
フォント	6	非対応	非対応	非対応	非対応
デバイスピクセル比	11	18	対応	対応	11.10
CPU のコア数	10	3.5	3[M]，4[C]	4	10.6
Canvas Fingerprinting (ブラウザ，OS の種類)	9	2	4[M]	3.1，3.2[C]	9，9.6[C]
Canvas Fingerprinting (GPU の種類)	11[M]	4	9[M]	非対応 5.1[M]，8[C]	12[M]
バッテリー状況	非対応	11，10[M]，10[C]	非対応	非対応	非対応
ハードディスク空き容量	非対応	非対応	対応	非対応	15.0.1147.130
ディスプレイのリフレッシュレート	10	4	10[M]	6.0.5，6[M]	15
カメラとマイクの数	非対応	非対応	対応	非対応	17.0.1241.45