Vol.2015-GN-93 No.31
Vol.2015-CDS-12 No.31
Vol.2015-DCC-9 No.31
2015/1/27

車載端末向け Web ブラウザの検討

松本貴士[†1]　近藤明宏[†1]　丸三徳[†1]

車載端末において HTML5 が注目されている．しかしながら，Web コンテンツの HTML5 化に伴い Web ブラウザが使用するメモリ量は増大する傾向にあり，Web ブラウザが動的に確保するメモリ使用量（ヒープ量）の増大は避けられない．PC やスマートフォンと比較してメモリ搭載量で劣る車載端末では Web ブラウザが他のソフトウェアに割り当てられた領域を破壊する等の他のソフトウェアやシステムの動作に影響を及ぼす可能性がある．本報告では，車載端末のスペックで動作する Web ブラウザを試作し，ヒープ量を抑制する方法およびヒープ量の削減に影響があるガベージコレクタの挙動に着目して評価を行った．また，評価結果から車載端末向けの Web ブラウザで必要となるメモリ破壊を起こさないための設計指針を示した．

Study on Web Browser for In-Vehicle Devices

TAKASHI MATSUMOTO[†1]　AKIHIRO KONDO[†1]
MITSUNORI MARU[†1]

HTML5 is interested in in-vehicle devices market. However, memory consumption of a web browser has increased because web contents adapt to HTML5. In addition, a web browser is unavoidable to bloat of heap memory size. Since amount of memory with in-vehicle devices is limited to that of PCs and smart phones, a web browser is possible to be crashed other softwares and systems because of destroying memory regions of them. In this research, we made a prototype web browser for in-vehicle devices and we evaluate control of heap memory consumption and behavior of garbage collection which is effect for decrease of heap memory size. Furthermore, we provide a design policy on in-vehicle device web browser to avoid destroying memory regions.

1. はじめに

従来のカーナビゲーションシステムをはじめとする車載端末向けネットワークサービスは，交通情報配信やオペレータサービス等の自動車向け情報サービス（テレマティクスサービス）の提供に特化して提供されてきた．ネットワークサービスはプレミアムサービスとして位置づけられ，付加価値として捉えられていた．しかしながら，スマートフォンの普及およびテザリングサービスの普及によって，車載端末によるネットワーク接続がベースラインの機能として認知されるようになった．

加えて，In-Vehicle Infotainment（IVI）と呼ばれるコンセプトの登場により，スマートフォン等で実現されている Web アプリケーション（Web アプリ）を車載端末の機能として統合することが求められるようになった．Web アプリの特徴は，Human-Machine Interface（HMI）に相当する機能を端末側で処理するとともに，それらと協調動作するサービスの実態をサーバ側で処理する点である．サーバ側の機能は端末側の機能と比較して修正や改変が容易であることを活用して，迅速で拡張性の高いサービス提供可能となる．現在では，車載端末によるネットワーク接続が容易になったために，ウェブアプリを車載端末上で提供することがエンドユーザやカーメーカから求められるようになった．

一方で，車載端末ではエンドユーザやカーメーカの要求に応えるために，ハードウェアや OS が異なる構成の上にアプリケーションを実装することも多く，開発期間の長期化が課題となっている．

このような背景の下，車載端末においても HTML5 [1]が注目されている．HTML5 は，HTML や JavaScript [a]，CSS で構成され，Web ブラウザ上で実行される．HTML5 に準拠する Web ブラウザであれば OS に非依存で同じ Web アプリや Web ページ（以下，Web コンテンツ）が動作するというクロスプラットフォーム性を有する．さらに，HTML5 で記述された Web コンテンツは，OS 上で直接実行可能なアプリケーション（ネイティブアプリ）と同等の機能や表現力を持つ[2]．ネイティブアプリとして提供してきた機能を Web コンテンツとして提供することで，アプリの開発コストを削減することが期待されている．

しかしながら，Web コンテンツも HTML5 によりネイティブアプリ相当の機能を提供するようになったことで高度化，複雑化しており，Web ブラウザが使用するメモリ量も増大する傾向にある．また，Web ブラウザでは JavaScript によるメモリリークおよびヒープ領域の肥大化も問題となっており，PC やスマートフォンと比較してメモリ搭載量で劣る車載端末への搭載には課題があることが予想される．

†1 (株)日立製作所　横浜研究所
Hitachi Ltd., Yokohama Research Laboratory

[a] JavaScript は，Oracle Corporation およびその子会社，関連会社の登録商標です．

Vol.2015-GN-93 No.31
Vol.2015-CDS-12 No.31
Vol.2015-DCC-9 No.31
2015/1/27

本報告では，車載端末のハードウェアスペックで動作する Web ブラウザを試作し，PC 等よりもメモリ搭載量で劣る車載端末における課題を解消するために，ヒープ量を抑制する方法およびヒープ量の削減に影響があるガベージコレクタの挙動に着目して評価を行った結果を報告する．また，評価結果から，車載端末向けの Web ブラウザで必要となるメモリ破壊を起こさないための設計指針を示す．

2. 先行研究

2.1 Web ブラウザの改良

比較的新しい技術であり機能や性能の向上が著しい Web ブラウザでは，Web ブラウザ自身の改良を目的とした研究が行われている．

ひとつは，頑健性向上を目的としたマルチプロセス化に関する検討である[3]．この中で，マルチプロセス化により Web コンテンツのロード時間が長くなること，メモリ使用量が増加することが示されている．また，JavaScript が動的に確保したヒープ領域を開放する機構であるガベージコレクタに関する研究も行われている[4]．ヒープ領域をオリジン単位で論理的に分割することで，ガベージコレクションの実行時間が削減できることが報告されている．一方で，JavaScript によるメモリリークおよびヒープ領域の肥大化とこれらを解消するためのガベージコレクションによる性能低下は課題となっており，メモリリークの検出に関する研究も行われている[5].

2.2 組み込み分野への適用

上記のような Web ブラウザそのものに関する検討に加えて，車載端末を含む組み込み分野への適用も検討されている．

車載端末を想定したハードウェアを用いた，C++と JavaScript の演算性能の比較評価がなされている[6]. また，デジタル TV への搭載を想定し，ブラウザが使用するコード量の削減方法や組込みシステムの安定性確保のためのプロセス分離機能についての報告がある[7].

3. 課題と目標

3.1 解決すべき課題

Web ブラウザでは，ブラウザ自身の機能向上のために，メモリ使用量が増大している．加えて，HTML5 化に伴い Web コンテンツもネイティブアプリ相当の機能を提供するようになったことで高度化，複雑化しており，コンテンツが消費するメモリ量も増大する傾向にある．PC 向け Web ブラウザのメモリ使用量を確認したところ，タブ 3 つの場合で 300MB 程度を使用しており[8]，PC やスマートフォンと比べてメモリ搭載量が劣る車載端末にそのまま搭載することは困難であることがわかった．

Web ブラウザのメモリ使用量が大きい理由は，コンテンツを高速に描画するためである．現在の Web ブラウザはいわゆるタブブラウザとして提供されており，新たな Web コンテンツをロードする際に新たなタブにそのコンテンツのインスタンスを生成することで，再表示時のロードを不要とし表示時間の短縮を実現している[3]．加えて，Web ブラウザでは，Web コンテンツをメモリ上に展開したままキャッシュとして利用し再表示時の高速化を実現している[9][10]．基本的には，新たなタブを追加するたびにキャッシュが追加されるため，メモリ使用量が増加することになる．また，先行研究にあるとおり JavaScript によるメモリリークおよびヒープ領域の肥大化とこれらを解消するためのガベージコレクションによる性能低下も課題となるが，そもそもの原因はコンテンツの JavaScript コードであり，Web ブラウザ自身による解消は困難である[5].

以上のように，Web ブラウザによるメモリ使用量の増大は複数の要因で構成され，現時点では根本的な解消が難しい．たとえば，Web ブラウザを母体として開発された Firefox OS [b]では，システム全体のメモリ使用量が低下すると OS 側でコンテンツが割り当たっているプロセスを終了することでメモリを開放する機能を備えている[11]．車載端末等の組み込みシステムでは，リアルタイム性の高い OS 上で，設計時にそれぞれのソフトウェアに使用可能なメモリ量の上限を割り当て，ソフトウェア自身がその範囲内で動作するように設計することが多い．Web ブラウザが動的に確保するメモリ領域（ヒープ領域）の増大は避けられないため，他のソフトウェアに割り当てられた領域を破壊する等，他のソフトウェアやシステムの動作に影響を及ぼす可能性が高い．

3.2 目標

本研究の目標は，車載端末のスペックで動作する Web ブラウザを試作し，PC 等よりもメモリ搭載量で劣る車載端末におけるブラウザによるメモリ破壊を避けるために，ヒープ量を抑制する方法およびヒープ量の削減に影響があるガベージコレクタの挙動に着目して評価を行こと．また，評価結果から，車載端末向けの Web ブラウザで必要となるメモリ破壊を起こさないための設計指針を示すことである．

[b] Firefox は、Mozilla Foundation の登録商標です．

Vol.2015-GN-93 No.31
Vol.2015-CDS-12 No.31
Vol.2015-DCC-9 No.31
2015/1/27

4. Web ブラウザの試作

4.1 ヒープの抑制

Web ブラウザが確保するヒープ領域には，主に以下が含まれる．

① タブ生成に起因するもの
② レンダリング過程で生成される中間データおよびコンテンツのインスタンス等 Web ブラウザがキャッシュとして保持するもの[9][14]
③ JavaScript が動的に確保するもの

①は Web ブラウザがタブを生成するたびに獲得されるメモリ領域であり，タブを無制限に生成するとヒープ量が増加し続けることになる．よって，本試作ブラウザでは生成可能なタブ数を 4 つとして，タブ生成に起因するヒープ量を抑制することにした．4 つに設定した理由は，以前検討したコンテンツの種類で起動するタブを変えるという考えによる[8]. 今回は, 常駐するコンテンツ用のタブ 3 つ（音楽再生，音声認識，ナビゲーション）および非常駐コンテンツ用のタブ 1 つを想定した．

②の Web ブラウザが保持するコンテンツ関連のデータに起因するヒープ量はロードされるコンテンツに依存することになり，Web ブラウザとしての制御は難しい．同様に③の JavaScript が確保するヒープ領域についてもコンテンツ次第だが, JavaScript により明示的に開放が行われない場合にはガベージコレクタが開放を行うため，任意のタイミングでヒープ量が減少することが考えられる．

4.2 試作 Web ブラウザの概要

試作した Web ブラウザに, Web コンテンツのロード時（例えば，Web コンテンツ中の URL へのリンクがクリックされた時や onclick=”location.href=’http://aaa.html’ ”が実行された時）に，起動したいタブの番号を指定し，タブの指定がない場合には，Web コンテンツのロードが要求されたタブで起動する機能を実装した．この機能を用いて，最大 4 つのタブに対して，意図的にロードするタブを変えたり，タブの数に上限を設けたりすることでタブ生成に起因するヒープ量の抑制を実現した．

また，ロードするタブを指定する機能は，JavaScript から使用できるように NPAPI [12]を用いた．レンダリングエンジンには WebKit [c] [13]を使用し，単一プロセスで動作するように実装した．

[c] WebKit は，Apple Inc.の登録商標です．

5. 評価

5.1 評価の概要

PC 等よりもメモリ搭載量で劣る車載端末では, ヒープ領域の増大とそれによる他のソフトウェアの領域の破壊が課題になるため，生成可能なタブ数を制限することによるタブ生成によるヒープ量（4.1 の①）の抑制の効果および JavaScript が確保するヒープ量（4.1 の③）の削減に影響があるガベージコレクタの挙動に着目して評価を行った．

試作した Web ブラウザでは, 生成可能なタブの数を 4 つに制限することで，ヒープ量を抑制することにした．この効果を確認するために，タブが 1 つの場合とタブが 4 つの場合とでヒープ量の差の確認により，タブ生成によるヒープの増加量を測定した．また，Web ブラウザではヒープ領域の増大が予想されるため，メモリ搭載量で劣る車載端末で必要となりそうなメモリ量を見積もるために，Web ブラウザが保持するコンテンツ関連のヒープ（4.1 の②）および JavaScript が確保するヒープ（4.1 の③）を増加させてヒープの増加量を測定した．続いて，ガベージコレクタが動作することによる JavaScript が確保するヒープ（4.1 の③）への影響を調査するために，長時間コンテンツを実行し，ヒープ量を増加させたときの変化を測定した．

5.2 評価システム

評価システムの構成を図 1 に示す. 評価用端末に試作した Web ブラウザを搭載する. Web ブラウザで表示するコンテンツは PC（Windows 7 [d]）内の HTTP サーバ（Apache HTTP server [e]）に保存されたものを取得する．評価用端末と PC はスマートフォン（iPhone5 [f]，iOS7 [g]）を介して接続した．評価用端末とスマートフォンは USB で，スマートフォンと PC は Wi-Fi [h]テザリングを使用して IEEE 802.11g で接続した. 通信遅延等の揺らぎの影響を減らすために，インターネット等の外部ネットワークは使用せずにこの構成とした．評価用端末の仕様は表 1 に示すとおりである．試作した Web ブラウザのコードサイズは ROM 約 20.0 MB, RAM 約 1.9 MB の合計 21.9 MB となった（表 2）.

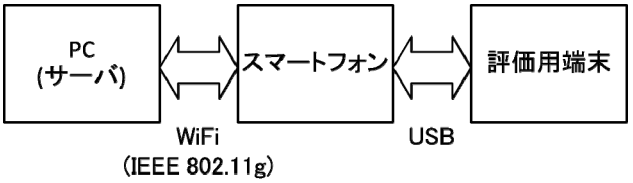

図 1 評価システムの構成

[d] Windows ならびに Windows Embedded Compact 7 は, Microsoft Corporation. の登録商標です.
[e] Apache は，Apache Software Foundation の登録商標です.
[f] iPhone は，Apple Inc.の登録商標です.
[g] iOS は，Cisco Systems, Inc.の登録商標です.
[h] Wi-Fi は，Wi-Fi Alliance の登録商標です.

Vol.2015-GN-93 No.31
Vol.2015-CDS-12 No.31
Vol.2015-DCC-9 No.31
2015/1/27

表 1　評価用端末仕様

CPU クロック	800MHz
メモリ	512MB
画面解像度	800 x 480 px
OS	Windows Embedded Compact 7

表 2 試作 Web ブラウザのコードサイズ

ROM[KB]	20,048
RAM[KB]	1,892
合計[KB]	21,940

5.3 ヒープ量の測定

ヒープ量の測定は, 試作 Web ブラウザのプロセスが動的に確保するヒープ領域を特定タイミングで観測し，その差をとることで実現した．ヒープ量の測定単位は 64 KB で，測定タイミングはコンテンツのロード完了直後とした．

Web ブラウザのヒープ領域は，タブ生成によるヒープ（4.1 の①），Web ブラウザが保持するコンテンツ関連のヒープ（4.1 の②）および JavaScript が確保するヒープ（4.1 の③）で構成される．今回の測定では，これらの変化を間接的に観測することになる．タブ生成によるヒープ（4.1 の①）はタブを生成したタイミングで変化量を測定すればよいが，Web ブラウザが保持するコンテンツ関連のヒープ（4.1 の②）および JavaScript が確保するヒープ（4.1 の③）についての内訳を判別することは難しい．また，JavaScript が確保するヒープ（4.1 の③）については，JavaScript により明示的に開放が行われない場合には，ガベージコレクタが開放を行うため，任意のタイミングで減少することが考えられる．

今回は, 車載向け Web ブラウザでの使用することが多くなると思われる WVGA サイズ（800 x 400 px）の画像を JavaScript でロードするコンテンツ（図 2）を使用して，ヒープ量を測定した．車載端末は，CPU スペックでは劣るが画像のデコードについてはハードウェアのアクセラレーションが期待できる．JavaScript や CSS による描画よりも単に画像を貼り付けたほうが高速に描画できる可能性が高いため，このコンテンツを採用した．また，このコンテンツでは，画像データ等の JavaScript で確保した資源の解放を行わないので，JavaScript が確保するヒープ（4.1 の③）はガベージコレクタにより随時解放されることになる. また，WVGA の画像を表示するページはそれぞれ別のディレクトリ，html として配置し，レンダリング時の Web ブラウザが保持するコンテンツ関連のヒープ（4.1 の②）へのヒット率を下げて新たにヒープとして作成されるようにした．

図 2　WVGA 画像を表示するページの例

5.4 タブ生成によるヒープ量の変化

タブ生成によるヒープ量（4.1 の①）の変化を評価するため, 4 つのコンテンツを 1 つのタブにロードした場合と，4 つのタブにロードした場合（表 3）での試作 Web ブラウザが確保したヒープ量を測定した. 使用したコンテンツは，リンク先を記述したページ（図 3）と WVGA の画像を表示するページ（図 2）を 3 つの合計 4 つである． リンク先を記述したページは，試作した Web ブラウザの URL 入力や Bookmark の呼び出し等の機能を備えていないという制約上必要になる．

測定結果を表 4 に示す．タブ 1 つは 4 つのコンテンツを Tab1 にロードした時のヒープ量，タブ 4 つは Tab1 にリンク先ページ，Tab2 に WVGA 画像を表示するページ 1（WVGA1），Tab3 に WVGA 画像を表示するページ 2（WVGA2），Tab4 に WVGA 画像を表示するページ 3（WVGA3）をロードした時のヒープ量である．

Web ブラウザ全体でのメモリ使用量は，タブ 1 つの場合約 64.8 MB，タブ 4 つの場合で約 70.6 MB であった．動的に確保されたヒープ量の合計はタブ 1 つの場合で約 42.8 MB，タブ 4 つの場合で約 48.6 MB となり，タブ 4 つの場合が約 5.8MB ほどヒープ量が大きくなった. 内訳を見ると，Tab を生成した WVGA1，WVGA2，WVGA3 のロード時にそれぞれ 1MB から 2.7MB 程度の範囲で増加している．1 つタブを生成ごとに，約 2MB 程度のヒープ量の増加があることを確認できた．

Vol.2015-GN-93 No.31
Vol.2015-CDS-12 No.31
Vol.2015-DCC-9 No.31
2015/1/27

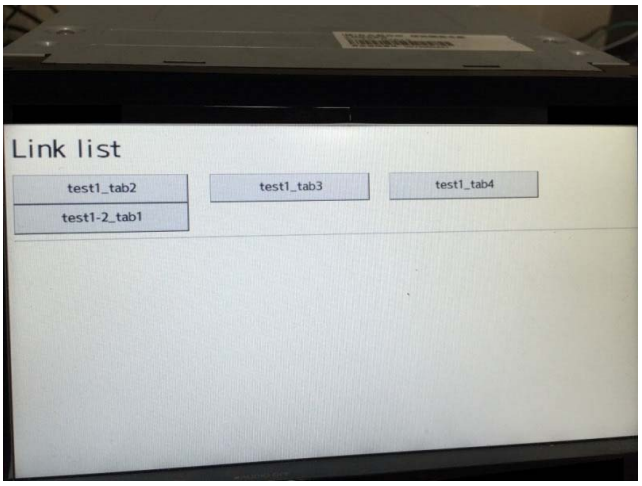

図 3　リンク先を記述したページの例

表 3　ヒープ量確認用コンテンツ

コンテンツ	タブ 1 つ	タブ 4 つ
リンク先を記述したページ	Tab1	Tab1
WVGA 画像を表示するページ 1	Tab1	Tab2
WVGA 画像を表示するページ 2	Tab1	Tab3
WVGA 画像を表示するページ 3	Tab1	Tab3

表 4　タブ生成によるヒープ量の変化

コンテンツ	タブ 1 つ [KB]	タブ 4 つ [KB]	差 [KB]
リンク先ページ	35520	35584	64
WVGA1	768	2880	2112
WVGA2	5824	6784	960
WVGA3	704	3392	2688
ヒープ量	42816	48640	5824
ブラウザ合計	64756	70580	-

5.5 車載端末で必要となるメモリ量

続いて，メモリ搭載量で劣る車載端末で必要となりそうなメモリ量を見積もるために，車載端末で多用されると思われるコンテンツを用いて Web ブラウザが保持するコンテンツ関連のヒープ（4.1 の②）および JavaScript が確保するヒープ（4.1 の③）を増加させてヒープの増加量を測定した．ヒープ量を増加させるために，異なる WVGA 画像を 30 回表示するコンテンツを用意し，タブ 1 つの場合は合計 90 枚の画像を表示するコンテンツ，タブ 4 つの場合は 30 枚の画像をそれぞれ 3 つのタブで表示するコンテンツを使用した．

表 5　合計 90 枚の画像を表示するコンテンツ

コンテンツ	タブ 1 つ	タブ 4 つ
リンク先を記述したページ	Tab1	Tab1
WVGA 画像を表示するページ 1 x 30	Tab1	Tab2
WVGA 画像を表示するページ 2 x 30	Tab1	Tab3
WVGA 画像を表示するページ 3 x 30	Tab1	Tab3

ヒープ量の測定結果を表 6 に，画像を 90 枚表示する間のヒープ量の推移を図 4 に示す．5.4 の画像を 1 枚ずつ，合計 3 枚の画像を表示した場合に比べて，ヒープ量が 20 MB 程度増加し，タブ 1 つの場合，タブ 4 つの場合双方ともブラウザ全体で約 88 MB のメモリを使用していることがわかった．車載端末で多用されると思われるコンテンツである WVGA 画像を 90 枚表示によってヒープ量が約 65 MB も増加していることから，車載端末に Web ブラウザを搭載して一般的な Web コンテンツを動作させるには，128MB 以上のメモリを割り当てる必要があると考える．

一方，タブ 1 つの場合とタブ 4 つの場合のヒープ量の差は 700 KB 程度となり，画像を 1 枚ずつ，合計 3 枚の画像を表示した場合に比べて差が縮小した． 図 4 の内容を確認すると，タブ 1 つの場合とタブ 4 つの場合双方とも 6 枚目の表示完了までヒープ量が増加した後，7 枚目の表示完了時にはヒープ量が減少している．その後も増減を繰り返しながらも徐々にヒープ量は増加していく．この間，タブ 4 つのヒープ量はタブ 1 つのヒープ量と比べておよそ 2.5 MB ほど大きい状態を継続するが，タブ 4 つのヒープ量は，61 枚目の表示完了後，63 枚目の表示完了までに大幅に減少し，タブ 1 つのヒープ量よりも小さくなる．その後，75 枚目の表示完了後にほぼ同じヒープ量になり，同じペースで増加した．

ヒープ量の減少は，ガベージコレクタによる JavaScript が確保するヒープ（4.1 の③）の開放が原因と考えられる．6 枚目の表示完了後にガベージコレクタが積極的に動作をはじめ，61 枚目の表示完了まではタブ生成に起因するヒープ量の影響が見られるが，それ以降は Web ブラウザが保持するコンテンツ関連のヒープ（4.1 の②）および JavaScript が確保するヒープ（4.1 の③）の影響が支配的になるためと推測する．

Vol.2015-GN-93 No.31
Vol.2015-CDS-12 No.31
Vol.2015-DCC-9 No.31
2015/1/27

表 6 WVGA 画像 90 枚表示時のヒープ量

コンテンツ	タブ 1 つ [KB]	タブ 4 つ [KB]	差 [KB]
リンク先ページ	35328	35328	0
WVGA1	14208	16640	2432
WVGA2	10496	10240	-256
WVGA3	5568	4096	-1472
ヒープ合計	65600	66304	704
ブラウザ合計	87540	88244	-

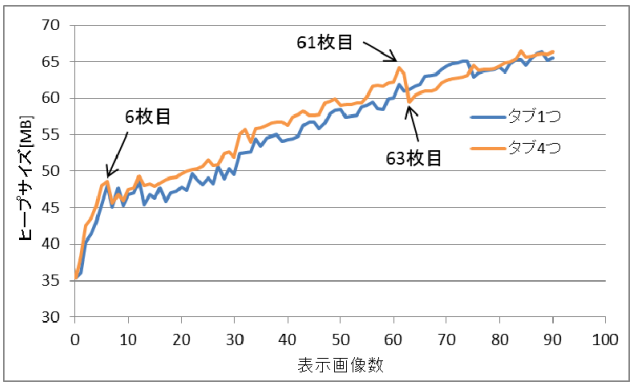

図 4　ヒープ量の推移

5.6 JavaScript が確保するヒープ量の変化

5.5 でガベージコレクタが積極的に動作をはじめてから一定期間経過すると Web ブラウザが保持するコンテンツ関連のヒープ（4.1 の②）および JavaScript が確保するヒープ（4.1 の③）が支配的になると予想した．これを確認するために，5.5 のタブ 1 つの評価で使用した WVGA 画像を 90 枚表示するコンテンツをループさせることでガベージコレクタを長期間動作させ，その時のヒープ量を観測した（図 5）．この結果から分かるように，長期間で見るとヒープ量は単調増加になった．

ヒープ領域の内訳を推定するために，ヒープ量の変化量（図 5 の凸凹の内訳，図 6）と変化量の最大値および最小値（表 7）を確認した．変化量の最大値は 2 枚目完了時の約 4.4 MB，減少量の最大値は 480 枚目完了時の約 4.6 MB であった．今回の試作 Web ブラウザおよびコンテンツでは，±4.5 MB 程度の振れがあることがわかった．ヒープ量は主に Web ブラウザが保持するコンテンツ関連のヒープ（4.1 の②）および JavaScript が確保するヒープ（4.1 の③）で構成されるため，1 周目（90 枚目まで）ではブラウザが確保するコンテンツ関連のヒープの影響が出ることを期待したが，変化量の差分には現れなかった．

続いて， 90 枚を表示するごとのブラウザのヒープ増加量の推移を確認した（図 7）．1 週目には 32 MB の増加があるが，2 週目以降は 15 MB から 20 MB の増加にとどまる．本実験のコンテンツは 90 枚の画像の表示のループ処理により同じものが繰り返し実行されることになるので，Web ブラウザが保持するコンテンツ関連のヒープ（4.1 の②）は 2 週目以降増加しないことが原因と考えられる．よって，単調増加の理由はガベージコレクタが JavaScript のヒープ領域を十分に開放し切れていないことによると推測する．以上から，長期間コンテンツを実行すると JavaScript が確保するヒープ（4.1 の③）が増加し続ける可能性があるといえる．

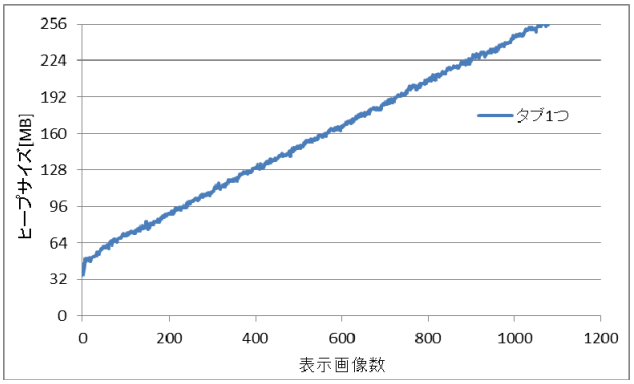

図 5 長期間のヒープ量の推移

表 7　ヒープ量の増減量

最大値(2 枚目完了時)	4,352 KB
最小値(480 枚目完了時)	- 4,608 KB

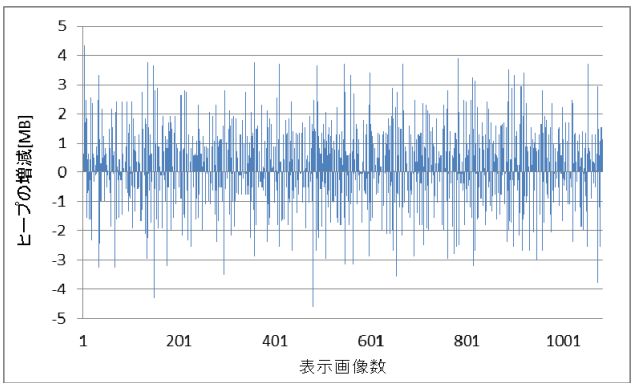

図 6　ヒープ量の増減推移

Vol.2015-GN-93 No.31
Vol.2015-CDS-12 No.31
Vol.2015-DCC-9 No.31
2015/1/27

図 7　90 枚表示毎のヒープ量増加量

5.7 考察

以上の測定により, 5.4 でタブ生成により約 2 MB のヒープ量の増加があること，5.5 で WVGA 画像を 90 枚表示させたときのブラウザ全体のメモリ使用量は約 88 MB であり，少なくとも 128 MB のメモリを割り当てる必要があること，5.6 では，長期間コンテンツを実行するとガベージコレクタが解放しきれないことにより JavaScript によるヒープ量が増加し続けることが分かった.

JavaScript が確保するヒープ（4.1 の③）については JavaScript のメモリリークが発生する可能性が高く, ヒープ量の確定が困難であると言える．Web ブラウザで使用可能なメモリ量が限られる場合には, JavaScript が確保するヒープ（4.1 の③）が多く確保できるよう，タブの生成数を制限すべきである．たとえば，Web ブラウザに割り当てられたメモリ量が 128 MB であれば，タブ数を 4 つとすると，22 MB（プログラム） + 8MB（タブ 4 つ） = 30 MB となり, ヒープ領域に 98 MB を割り当てることができる. また，256MB 以上の割り当てが可能な状況であれば，タブを 20 個開いても 40 MB 程度となり影響は小さいため，タブの生成数に上限を設ける必要はない.

6. まとめ

PC 等よりもメモリ搭載量で劣る車載端末では, ヒープ領域の増大とそれによる他のソフトウェアの領域の破壊が課題になるため，生成可能なタブ数を制限することによるタブ生成によるヒープ量の抑制の効果および JavaScript が確保するヒープ量の削減に影響があるガベージコレクタの挙動に着目して評価を行った.

また，評価結果から，車載端末向けの Web ブラウザで必要となるメモリ破壊を起こさないための設計指針を示した.

参考文献

[1] HTML5 A vocabulary and associated APIs For HTML and XHTML
http://www.w3.org/TR/html5/
[2] Antero Taivalsaari and Tommi Mikkonen, “The Web as an Application Platform: The Saga Continues”, IEEE 37th EUROMICRO Conference on Software Engineering and Advanced Applications, 2011.
[3] Charles Reis and Steven D. Gribble, “Isolating Web Programs in Modern Browser Architectures”, Proceedings of the 4th ACM European conference on Computer systems, 2009.
[4] Gergor Wagner, Andreas Gal, Christian Wimmer, Brendan Eich, and Michael Franz, “Compartmental Memory Management in a Modern Web Browser”, Proceedings of the International Symposium on Memory Management, pages 119-128. ACM, 2011.
[5] Jacques A. Pienaar and Robert Hundt, “JSWhiz: Static analysis for JavaScript memory leaks”, Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), 2013.
[6] S. Isenberg, M. Goebl, and U. Baumgarten, “Is theWeb Ready for In-Car Infotainment? A Framework for Browser Performance Tests Suited for Embedded Vehicle Hardware”, 14th IEEE International Symposium on Web Systems Evolution (WSE), 2012.
[7] 平野裕, 深井祐介,” Web ブラウザエンジン WebKit の映像製品への適用”, 東芝レビュー Vol.67, No.8, 2012.
[8] 松本貴士, 近藤明宏, 丸三徳, “車載向け HTML5 アプリのライフサイクル管理方法の提案”, 情報処理学会研究報告 コンシューマデバイス&システム(CDS), 2014
[9] MemoryCache - Webkit Wiki
http://trac.webkit.org/wiki/MemoryCache
[10] WebKit Page Cache I – The Basics
https://www.webkit.org/blog/427/webkit-page-cache-i-the-basics/
[11] Out of memory management on Firefox OS
https://developer.mozilla.org/en-US/Firefox_OS/Platform/Out_of_memory_management_on_Firefox_OS
[12] NPAPI – Mozilla Wiki
https://wiki.mozilla.org/NPAPI
[13] The WebKit Open Source Project
https://www.webkit.org
[14] How WebKit Loads a Web Page
https://www.webkit.org/blog/1188/how-webkit-loads-a-web-page/