社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Webフロントエンド開発者のための ユーザ参加型エラー収集システムの提案

山本　将弘†　　柗本　真佑†　　楠本　真二†

† 大阪大学大学院情報科学研究科
〒 565–0871 大阪府吹田市山田丘 1–5
E-mail: †{m-yamamt,shinsuke,kusumoto}@ist.osaka-u.ac.jp

あらまし　Web のフロントエンドを構成する HTML や JavaScript では様々なエラーが発生する．開発者はこれらのエラーの発生状況を早期に把握し，修正する必要がある．本研究では，エラーを早期に検知し改善に役立てることを目的として，参加型センシングのコンセプトを取り入れたエラー収集システムを提案する．エラー収集をユーザの協力により実施することで，多様な環境からの収集を実現できる．本稿では，Web において発生するエラーに関する背景と提案システムの概要，及び被験者実験により収集したエラーの事例について紹介する．
キーワード　Web フロントエンド開発，エラー収集，参加型センシング

1. はじめに

Web は現代における情報発信の基本となりつつある．従来の Web のフロントエンド（Web ブラウザ上に表示される UI のこと）は，Web ページと呼ばれるような HTML や CSS 等の静的なコンテンツのみで構成されるものが代表的であった．一方，近年では，Web アプリケーションと呼ばれるようなユーザからのインタラクティブな操作を想定する，よりリッチかつ複雑なシステムへと移り変わりつつある．この流れは，Web フロントエンド開発の複雑化を意味しており，その開発の支援は重要な課題であるといえる．

Web フロントエンド開発における難しさの一つは，クライアント側で発生するエラーの再現である．HTML や CSS，JavaScript 等の Web のリソース（以降，Web リソースと呼ぶ）の特徴の一つは，W3C や WHATWG，Ecma インターナショナル等の標準化団体が定めた中立的な仕様に対し，様々な Web ブラウザの実装が存在する点にある．この特徴は Web の中立性を保つ重要な性質である一方で，Web リソースの解釈がブラウザごとに異なるというブラウザ互換の問題の原因となる．近年では HTML5 の仕様策定に伴い，ブラウザ互換の問題は減少しつつあるものの，ブラウザごとに外観が変わるような問題も依然として存在している [1]．そのため，フロントエンド開発者は自身の開発した Web リソースに対し，様々なブラウザを用いて正しく動作するかを確認する必要がある．

フロントエンド開発のもう一つの難しさは，Web リソースが時間と共に「劣化する」という点である．Web の仕様は常に進化しており，過去に正しく動作していたページが動作しなくなるということも少なくない．例えば，SSL 保護されたページ内から保護されない HTTP 通信を呼び出す混在コンテンツという通信方法は，2013 年 8 月ごろから各ブラウザで警告から禁止に変更された．これにより，混在コンテンツを利用していた Web ページは，仕様変更後に開発者の期待通りに動作しなくなった [2]．これは，仕様やブラウザに対するリソースの相対的な劣化と見なすことができる．また，仕様の変更と同様にライブラリのアップデートも頻繁に行われている．これらのことから，開発を終えたリソースに対しても定期的なメンテナンスが必要不可欠である．

三つ目の難しさは，エラーに対する修正方法が把握しにくいという点である．Web フロントエンドの動的な責務を担う JavaScript は，他の言語と比べて柔軟性が高い．そのため，ある特定の機能を実現する場合であっても様々なコードの書き方が存在する．これはエラーの修正方法が多岐に渡るという課題に繋がる．StackOverflow(注1)等の良質なナレッジサービスも登場しているが，見つかったコードスニペットが期待する実行環境やバージョンと合致しているかを見極める必要もある．このスニペットのバージョン齟齬という問題は，Java や C，Python 等の他の言語でも同様に発生するが，前述したように Web の技術は仕様の変化が激しいことからも，古い情報がノイズになりやすいといえる．

本研究では，Web フロントエンド開発におけるエラーの発見および修正の効率化を目的として，参加型センシングのコンセプトを取り入れたエラー収集システムを提案する．参加型（Participatory または Voluntary）センシングとは，ユーザの自発的な参加を元に，計測対象のデータを広く回収する考えである [3]．提案システムではこの考えを取り入れた 2 つの機能を実現する．まず機能 1 では，様々なユーザに自主的に協力し

(注1)：https://www.stackoverflow.com

てもらい，Web 上に存在する膨大な数の Web リソースを対象としたエラー発生報告の回収を行う．ユーザ参加型とすることで，様々なブラウザや実行環境の組み合わせが実現でき，さらにエラーの早期発見に繋がると考えられる．この機能 1 は先に述べた 2 つの問題（エラーの再現，リソースの劣化）の解決策となる．機能 2 では，収集したエラー情報に基づいてエラーの発生している Web ページを定期的に確認し，エラーの修正事例を回収する．これにより様々なエラー修正事例の提示が可能となる．

本稿では特に機能 1 に着目し，Web リソース上で発生するエラーに関する背景や，提案システムの概要，およびプロトタイプとして開発したシステムについて紹介する．さらに，被験者実験により得られたエラーの事例について紹介する．本稿の貢献は以下のとおりである．

- Web フロントエンド開発の支援を目的とした，ユーザ参加型のエラー収集システムを提案した．
- 収集価値，およびプライバシーの 2 つの観点から収集対象とするエラー情報の検討を行った．

2. Web の構成要素と参加型センシング

2.1 Web リソース

Web のフロントエンドは様々なリソースによって構成される．大別して，HTML や JavaScript 等のテキスト系のリソースと動画や画像等のマルチメディア系のリソースに分類することが可能である．本稿では，Web ブラウザ上で解釈，実行されるテキスト系のリソースに着目し，これらを単に Web リソースと呼ぶこととする．

Web リソースは他の一般的なプログラミング言語と異なり，コードオンデマンドと呼ばれるアーキテクチャに従って処理，実行される．コードオンデマンドでは，まずクライアント（一般的に Web ブラウザ）がサーバに対して HTTP リクエストを送信し，サーバから Web リソースを取得する．取得された Web リソースはクライアント側で解釈され，実行される．リソースのコンパイルや実行がクライアントサイドで行われるという点が特徴的であり，多種多様な実行環境を想定する必要がある．

2.2 Web リソース上で発生するエラー

前節で説明したように，Web リソースはクライアント側で実行される．そのため，ユーザ側の環境で様々なエラーが発生する．JavaScript のエラーの実情を調査した研究 [4] によると，アクセス数上位の多くのユーザが利用する Web サイトでも，平均で 1Web サイト当たり 4 個のエラーが含まれている．これらのエラーには，型に関するエラーや未定義変数の利用などの実行時エラーだけでなく，構文エラーのような開発途中に発見しやすいエラーも含まれている．

具体的なエラーの事例を示す． 図 1 は，ある Web ページ(注2) を Google Chrome で閲覧した画面のスクリーンショットである．同様に，同ページを Internet Explorer 6 で閲覧した画面

(注2)：http://bulletphysics.org/wordpress/

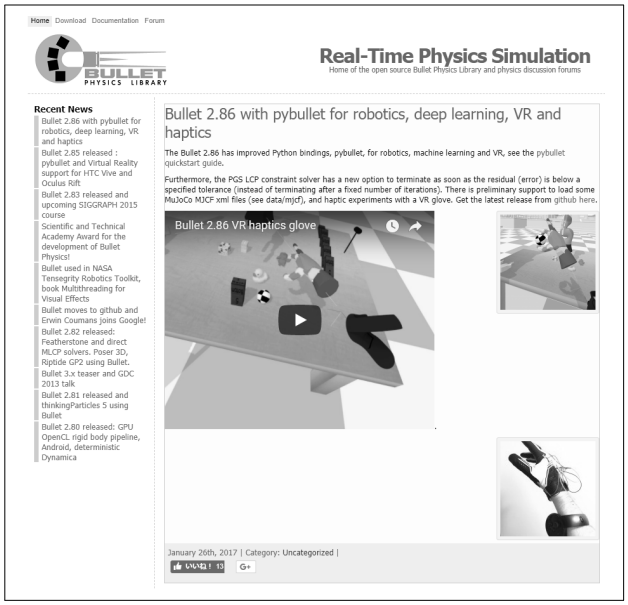

図 1　Google Chrome による Web ページの閲覧

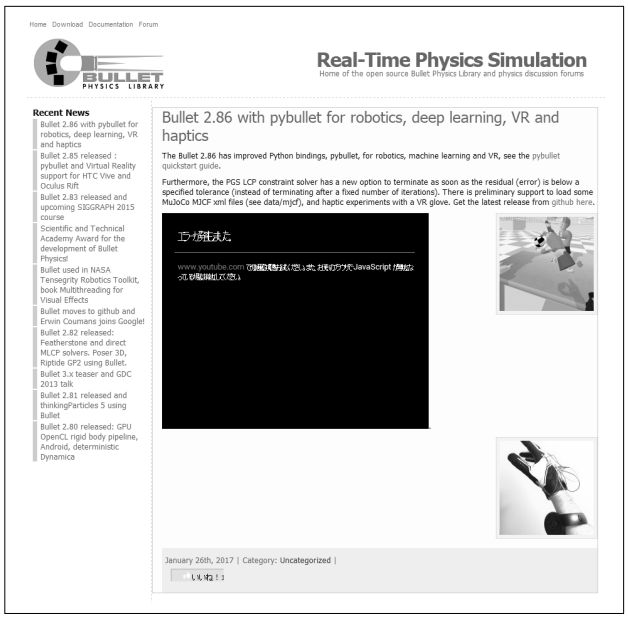

図 2　Internet Explorer 6 による Web ページの閲覧

```
Uncaught TypeError: Cannot read property 'msie' of undefined
    at HTMLDocument.<anonymous> (bulletphysics.org/:749)
    at c (jquery.js:3)
    at Object.fireWith [as resolveWith] (jquery.js:3)
    at Function.ready (jquery.js:3)
    at HTMLDocument.q (jquery.js:3)
```

図 3　Web ページ内部で発生していたエラー

を図 2 に示す．図 2 では画面中央の YouTube 動画や，画面下部の Facebook のいいねボタンが正しく表示されていない．加え，YouTube 画面の「エラーが発生しました」というエラーメッセージ自体もレイアウトが崩れている．

このエラーの原因は，jQuery ライブラリのアップデートに伴うものであると推測できる．Chrome で閲覧した際の JavaScript のコンソールでは，図 3 のようなエラーが発生している．具体的なエラー発生箇所（bulletphysics.org の 749 行目）のソースコードは以下の通りである．

```
// IE6 max-width for images
if (jQuery.browser.msie &&
 /MSIE 6\.0/i.test(navigator.userAgent) &&
 !/MSIE 7\.0/i.test(navigator.userAgent)) {
```

当該コードは，IE6 に対する画像サイズの特殊な処理を行っている部分である．この if 分岐内で利用されている `jQuery.browser.msie` というプロパティは，ブラウザが IE かどうかの判定結果が格納されるが，jQuery のバージョンアップに伴い削除された．これらの点から，この Web ページの開発者は IE6 利用者を想定して開発し，その後不用意に jQuery のバージョンを更新してしまったため，IE6 対応のコード自体が期待通りに動かなくなったと推測できる．

この事例のように Web リソースは，ブラウザの違いや仕様の変更，ライブラリのアップデート等の様々な要因でエラーが発生する．より品質の高い Web リソース開発のためには，まずエラーの発生状況を早期に把握し，その修正事例をいかに提示するかが重要となる．

2.3 参加型センシング

参加型センシングとは，ユーザの自発的な参加の元，ユーザの持つモバイル端末をセンサデバイスと捉えることで，広域かつリアルタイムなセンシングを実現するアイデアである [3] [5]．モバイル端末の普及を背景に考案され，近年の IoT の考え方が一般社会に広がるにつれてますます注目を集めている．

参加型センシングに関する研究をいくつか紹介する．Prabal らは大気汚染に関する情報を参加型センシングにより収集している [6]．また，Pengfei ら [7] はバスの待ち時間に関する研究を行った．ユーザの GPS 情報を用いることで到着時間を予想するシステムを提案している．

3. 提案システム：エラーリポジトリ

3.1 キーアイデア

本研究では，Web リソース上で発生するエラーの早期発見を目的として，エラーリポジトリというシステムを提案する．エラーリポジトリは，Web リソース上で発生するエラーの情報を収集する．エラーリポジトリのキーアイデアは参加型センシングの考え方をエラー収集に取り入れる点である．

参加型センシングのコンセプトを取り入れるにあたって，構成要素を置き換える．まず，ユーザの現在地は地点ではなく閲覧している Web ページに対応づけられる．センサデバイスはモバイル端末ではなく Web ブラウザに対応づけられる．また，センサ情報はエラーの情報やユーザ環境の情報である．

3.2 システムの概要

エラーリポジトリは 2 つの機能を提供する．これらの機能を実現するために，エラー送信プラグイン（Web ブラウザ拡張プラグイン）と収集サーバを作成する．

（1） エラー参加型のエラー収集・一元管理

（2） エラー解決時の差分を修正例として収集

前者を機能 1，後者を機能 2 と呼ぶことにする．それぞれの機能について説明する．

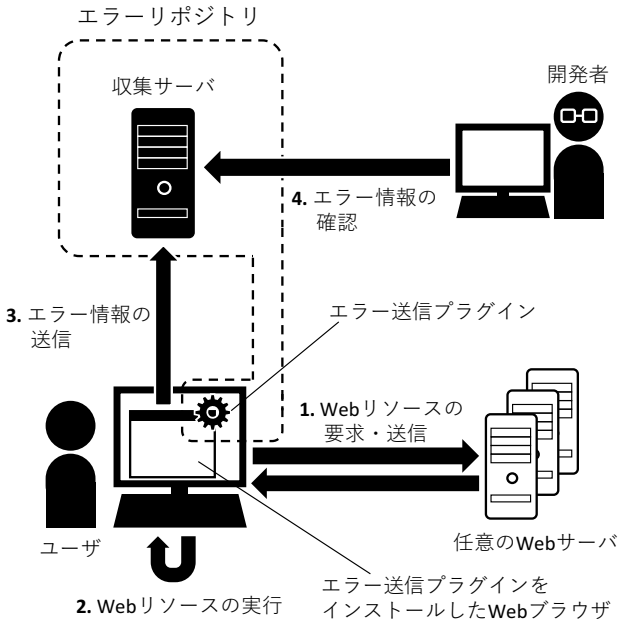

図 4 エラーリポジトリ機能 1 の仕組み

エラーリポジトリ機能 1 の動作の流れを図 4 に表す．機能 1 はユーザの協力を必要とする．ユーザには普段使用している Web ブラウザにエラー送信プラグインをインストールしてもらう．2.1 節で説明したコードオンデマンドという特性により，ユーザは自身の環境でソースコードを実行する．具体的には，閲覧したい Web ページを有する Web サーバに対して Web リソースを要求して受信する（図中矢印 1）．その後，受信した Web リソースをユーザ自身の環境で実行する（図中矢印 2）．実行時にエラーが発生すると，エラー送信プラグインがこれを検知し，収集サーバに対して情報を送信する（図中矢印 3）．収集サーバでは送信されてきたエラー情報をデータベースに保存して一元管理する．これらの情報はリアルタイムに更新されて，開発者はそれらのエラー情報を確認することができる（図中矢印 4）．この機能により，Web フロントエンド開発者はユーザ環境で起きたエラーを早期に発見できる．また，エラーが発生した環境などの情報も取得することができる．

機能 2 では，エラー収集から修正事例を収集する．機能 1 においてユーザからエラー情報を受信してから，一定の時間ごとにエラー情報内の Web リソースを取得して保存・実行する．ある時点で今まで発生していたエラーが発生しなくなったとき，前回と今回のソースコードの差分を抽出し，それらをエラーの修正例として収集する．

以降では，研究の第一歩として特に機能 1 について着目する．

3.3 エラーに関連する情報と収集の方針

収集するエラー情報について，エラー発生時の状況を表す項目を表 1 にまとめる．各項目とその一例を左側の 2 列に挙げている．右側の 3 列については，のちの収集の方針にて述べる．これらの項目の中で収集する項目を決定しなければならない．

収集すべき項目を判断する基準は，

（1） エラーの原因解明に役立つこと

（2） プライバシーを可能な限り守ること

表 1　収集可能なエラー情報の項目一覧と収集方針

項目	具体例	原因解明に役立つか	オプトアウトか	収集するか
URL	http://www.example.com/main.html	✓	✓	✓
エラーメッセージ	Uncaught SyntaxError: Unexpected token {	✓	✓	✓
スタックトレース	sub.js:5 main.js:7	✓	✓	✓
ブラウザ・OS	Google Chrome/61.0 Windows NT 6.1; x64	✓	✓	✓
日時	2017-08-20 15:30	✓	✓	✓
cookie	session_id=X9Qj5Bp	✓	✓	×
位置情報（GPS）	34.8, 135.5	✓	×	-
バッテリー残量	0.78	×	-	-

の 2 点である．この 2 つの基準はほぼ背反の関係にある．例えば，エラーの原因解明のためには Web ページの情報の全項目を収集するのが簡単であるが，その Web ページが個人情報を含む場合，プライバシーを侵害してしまうことになる．反対に，Web ブラウザの種類や URL などの項目を個人情報とするならば，原因解明のために収集できる項目がなくなってしまう．よって，これらの基準を考慮しながら，条件を満たすか満たさないかでフィルタリングをしながら収集する項目を決定する．

まず，エラー情報の各項目がエラーの原因解明に役立つかどうかでフィルタリングする．Web ブラウザを使用している端末のバッテリー残量などはエラーの原因と直接関係ないと考えられるので選択肢から外す．表中の他の項目は全て原因解明に役に立つ場合がある．URL はどの Web ページでエラーが起きたかを判断できるので役立つ．エラーメッセージとスタックトレースはエラーが発生した箇所を確認できるので役立つ．ブラウザ・OS の項目は環境に依存したエラーである場合にエラーが起きた環境を判断できるので役立つ．日時はエラー発生の状況を時間方向に振り返ることができるので役立つ．cookie はユーザ専用の Web ページにおいてエラーが発生した場合に状況を再現することができるので役立つ．位置情報（GPS）は，特定の区域（大学内など）でのみ発生するエラーに対して，その地域に限定したファイアウォールといった原因が検討できるので役立つ．

次にオプトアウトであるかどうかでフィルタリングする．オプトアウトとは，ユーザが拒否しない限り取得してもよい項目であることを表す．類似した用語にオプトインがあり，これはユーザが許可しない限り取得してはいけない項目である．プライバシーに関する項目はオプトインになることが多い．本研究ではオプトアウトの項目のみを収集することにする．エラーの原因解明に役立つ項目のうち，オプトアウトでないのは位置情報（GPS）である．よって位置情報（GPS）は収集する項目から外す．

フィルタリングされなかった全ての項目を収集したいところであるが，cookie は収集しないことに決めた．その理由は，cookie を知ることによってサービスのログイン乗っ取りができてしまう可能性があるためである．以上より，URL，エラーメッセージ，スタックトレース，ブラウザ・OS，日時の項目を収集することに決定した．

```
window.onerror = function(msg) {
  console.log(msg); //エラー発生時に行う処理
}
```

図 5　`window.onerror` の例

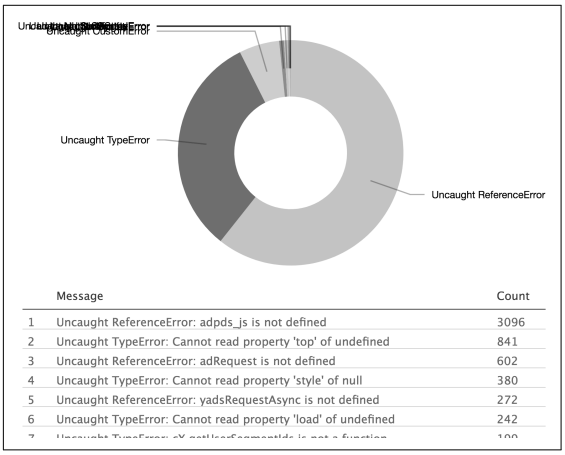

図 6　エラー情報公開ページ

3.4　エラーの収集方法

エラーを収集するためには，エラー送信プラグインが Web リソース上で発生した実行時エラーを検知できなければならない．これには `window.onerror` というイベントハンドラを利用する．`window.onerror` は実行時エラーが発生した時に呼び出され，図 5 のように開発者自身が独自の実装を行う．Google Chrome を始めとした多くの Web ブラウザがこの `window.onerror` の仕組みを取り入れている．エラーリポジトリでは，`window.onerror` の中で前節で選択した項目からなるエラー情報を非同期通信 Ajax を用いて収集サーバに送信する．

4.　システムの試作

本稿では実験に向けてエラーリポジトリの機能 1 のプロトタイプを作成した．収集サーバの実装については，メイン処理部分に Node.js を，データベース部分に MongoDB を用いた．また，エラー収集プラグインとして Google Chrome 用のブラウザ拡張機能と Firefox 用のプラグインをそれぞれ JavaScript で実装した．

また，収集したエラー情報を開発者に向けて公開するために

Web サーバを収集サーバのプログラム内に実装した(注3)．図 6 にスクリーンショットを示す．現時点では，収集したエラー情報のうち，エラーメッセージの種類の割合や送信された数の多いエラーメッセージが確認できる．

HTTPS 通信により閲覧している Web ページからはプラグインにおいてもデータの送信は HTTPS に限定される．エラー情報を送信する際のプライバシーも向上するため，エラーリポジトリの全ての送信は HTTPS 通信で行うようにした．

5. 実　　験

5.1 実験の目的

エラーリポジトリという Web リソース上で発生するエラーを収集するシステムを提案するにあたり，プロトタイプによる実験を行なって評価する．実験の目的を以下の 2 点に定める．

- 価値のあるエラー情報が収集できるか調べること
- プライバシーに関する情報が含まれるか調べること

価値のあるエラー情報とは，Web ページに影響をもたらすエラーであり，かつエラー情報を確認した時に原因が解明できるエラーであるとする．収集したエラー情報の中に価値のあるエラー情報が含まれていれば，エラーリポジトリの存在が意味を持つ．

3.3 節でエラー情報の項目の収集方針においてプライバシーに関する情報を考慮したが，実際に収集すると決めた項目にプライバシーに関する情報が含まれていないのかどうかを確認する．

5.2 実 験 設 計

実験において，被験者に対しては普段使用している Web ブラウザにエラー送信プラグインをインストールしてもらう．また，収集サーバを実際に稼働させて，エラー情報を受信・保存する．

被験者は 8 人であり，内訳は Google Chrome の使用者が 7 名，Firefox の使用者が 1 名である．実験を行う期間は 2017 年 8 月 15 日から 2017 年 8 月 31 日までの約 2 週間である．

6. 実 験 結 果

6.1 集計の結果

期間内に収集したエラー情報を集計した結果を述べる．まず，被験者から受信したエラー情報の総数は 4,172 件となった．これは，1 日あたり平均 245 件のエラー情報が収集サーバに送信されていることを表す．エラーメッセージの種類ごとの割合は図 7 のグラフのようになった．参照に関するエラーである ReferenceError が最も多く．次いで型に関するエラーである TypeError が多かった．

送信されたエラー情報を確認すると，日時以外の項目が一致する重複したエラー情報が多数見られた．そこでエラーメッセージと URL についてユニークな数を調べてみると，ユニークなエラーメッセージの数は 192，ユニークな URL の数は 481 となった．

(注3)：https://tyr.ics.es.osaka-u.ac.jp/error-collect/にて公開

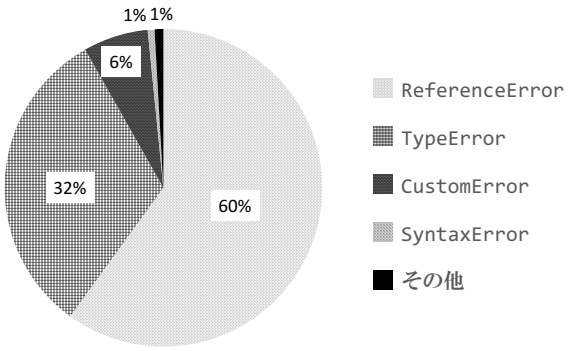

図 7　エラーメッセージの種類の割合

送信されたエラー情報について，実際に Web リソースの該当箇所を目視で確認した．その結果，価値のあるエラー情報が見つかったので，6.2 節でいくつか紹介する．また，ReferenceError の中で件数が多かったエラーメッセージに `adpds_js` や `adRequest` という変数が定義されていないというメッセージが含まれていた．これらについて，該当箇所を確認するなど調べてみると広告を非表示にするプラグインをインストールしている場合に発生するエラーであることがわかった．これらのエラー情報は，Web ページに影響をもたらすエラーであり，かつエラー情報を確認した時に原因が解明できるエラーであるが，価値のあるエラーとみなすかどうかは今後検討する必要がある．

6.2 収集できたエラーの事例

目視により Web リソースの該当箇所を確認すると，いくつかの事例で原因を判断することができた．2.2 節で紹介したエラーもその中の 1 つである．継続的にエラーの発生状況を調べるエラーリポジトリだからこそ発見できたエラーであるといえる．その他に見つかった価値のある事例を紹介する．

`Uncaught SyntaxError: Unexpected token {`

このエラーは，ソフトウェア工学関連の国際会議の Web サイト(注4)で発生していたエラーである．このエラーが発生した Web リソースの該当箇所を確認すると以下のようなソースコードを発見した．

```
<script src="bootstrap/3.3.0/css/bootstrap.min
.css"></script>
```

HTML の `script` タグにおいてエラーが発生している．`script` タグは JavaScript 等のスクリプト言語リソースをロードするために用いられる．しかしながら，この事例では `script` タグを用いて CSS ファイルを読み込もうとしている．そのため，ブラウザは CSS ファイルを JavaScript と見なして実行するため，CSS のブラケットの部分で構文エラーを発生させている．CSS は本来 `link` タグによって読み込むのが正しい文法であり，当該部分は開発者の期待通りに動作していないといえる．

また，同じエラーメッセージの別件を調べると別の Web サイト(注5)において，以下のようなソースコードも発見できた．

(注4)：https://icsme2017.github.io/
(注5)：https://mj-news.net

```
} elseif (strpos($site_rocal,'xxxxxx') !==
false){
```

このソースコードの記述は JavaScript の一部分である．PHP には `elseif` 構文が存在するが，JavaScript では `else` と `if` の間にスペースを挟まなければならない．そのため，構文エラーが発生した例である．

6.3 プライバシーに関する考察

実験の結果を目視で確認した結果，プライバシーに関する情報が含まれることが分かった．含まれていた箇所は以下のとおりである．

- URL のクエリ部分
- URL のパス部分
- エラーメッセージの一部分

URL のクエリ部分では ID やキーワードが含まれた．ID により個人が特定できる場合，プライバシーの情報がエラー情報に含まれていたことになる．場合によってはキーワードでも個人が特定されてしまう可能性もある．

```
http://www.example.com/main?id=xxxxxxx
```

```
http://www.example.com/search?q=xxxxxx
```

URL のパス部分やエラーメッセージに ID が含まれることがあった．下の例は Google Drive におけるエラー情報の URL やエラーメッセージである．サービスによっては，URL をメール等で交付することで，URL を知っている人々に限定して公開する場合がある．これらをエラー情報として公開してしまうと，限定でなくなるためサービスの意図に反する可能性がある．

```
https://drive.google.com/drive/folders/xxxx-
xxxxxx
```

```
Uncaught CustomError: Team Drive not found:
xxxx-xxxxxx
```

エラー情報を一般に公開する場合，これらのようなプライバシー情報を非公開または匿名化する必要がある．URL のクエリ部分以外にプライバシーの情報が含まれたのは Google Drive のみであった．そのため，公開する情報においてクエリ部分は常に非公開にし，Google Drive といった特定のサービスだけ独自のフィルタで非公開にするのが妥当と考えている．

7. 既存のエラー収集サービスとの比較

Web リソース上のエラー収集にはすでにいくつかのサービスがある．Errorception [8] や Airbrake [9] などが一例である．これらのサービスでも実行時に発生するエラーは window.error によって検知する．`window.onerror` 内ではサービスが運用するサーバに対してエラー情報を送信する．開発者は `window.onerror` を含むサービスが指定したソースコードを自身が開発している HTML のヘッダ部分に埋め込むことでサービスの恩恵が受けられる．

これらのサービスと本研究で提案したエラーリポジトリの決定的な違いは，エラー発生時にエラー情報を送信するコード片が Web リソース内に埋め込まれているか，Web ブラウザに拡張プラグインとして埋め込まれているかである．Web リソース内に埋め込む仕組みでは，エラーの発生を知ることができる対象となる開発者が限定される．一方，本研究のエラーリポジトリのようにプラグインとして Web ブラウザ側に埋め込むことができれば，ユーザが閲覧した全ての Web ページを開発した開発者がシステムの恩恵を受けることになる．ただ，新たに発生する問題として，ユーザに参加してもらうためにどのような動機付けを行うかについて考慮する必要がある．

8. おわりに

本稿では，Web フロントエンド開発におけるエラー収集システム，エラーリポジトリを提案した．さらに，エラーリポジトリのプロトタイプを作成し，被験者を対象とした実験を行った．収集したエラーをもとに Web ページを閲覧することで価値のある事例を探して，いくつかを紹介した．また，エラー情報の公開に向けて課題となるプライバシーに関する情報が含まれる場所について述べた．

今後は，見つかったエラーの事例について実際に開発者にコンタクトを取り，開発者にとって価値のあるエラーであるかを確認する予定である．また，エラーリポジトリのもう 1 つの機能であるエラー収集事例の収集について，実装・評価を行う．

謝　辞

本研究は，日本学術振興会科学研究費補助金基盤研究（S）(課題番号：JP25220003)，および文部科学省研究費補助金若手研究（B）（課題番号：JP26730155）の助成を得て行われた．

文　献

[1] A. Mesbah and M.R. Prasad, “Automated cross-browser compatibility testing,” International Conference on Software Engineering, pp.561–570, 2011.
[2] P. Chen, N. Nikiforakis, C. Huygens, and L. Desmet, “A dangerous mix: Large-scale analysis of mixed-content websites,” Information Security, pp.354–363, Springer, 2015.
[3] J.A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M.B. Srivastava, “Participatory sensing,” Center for Embedded Network Sensing, pp.1–5, 2006.
[4] F.S. Ocariza Jr, K. Pattabiraman, and B. Zorn, “Javascript errors in the wild: An empirical study,” International Symposium on Software Reliability Engineering, pp.100–109, 2011.
[5] 青木俊介，岩井将行，瀬崎薫，“参加型環境センシングを用いた統計情報構築のためのプライバシー保護手法，” 電子情報通信学会論文誌 B，vol.97，no.1，pp.41–50，2014.
[6] P. Dutta, P.M. Aoki, N. Kumar, A. Mainwaring, C. Myers, W. Willett, and A. Woodruff, “Common sense: Participatory urban sensing using a network of handheld air quality monitors,” The ACM Conference on Embedded Networked Sensor Systems, pp.349–350, 2009.
[7] P. Zhou, Y. Zheng, and M. Li, “How long to wait?: predicting bus arrival time with mobile phone based participatory sensing,” International Conference on Mobile Systems, Applications, and Services, pp.379–392, 2012.
[8] Rakesh Pai, “Errorception,” https://errorception.com.
[9] Airbrake, “Airbrake,” https://airbrake.io.