修士学位論文

題目

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

指導教員
楠本　真二　教授

報告者
山本　将弘

平成 30 年 2 月 7 日

大阪大学 大学院情報科学研究科
コンピュータサイエンス専攻

平成 29 年度 修士学位論文

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

山本　将弘

内容梗概

Web のフロントエンドを構成する HTML や JavaScript のような Web リソース上では，ブラウザで実行される際に様々なエラーが発生する．構文誤りが原因で発生する単純なエラーもあれば，ブラウザの仕様変更に伴って発生するような時間方向の要因を含むエラーも存在する．これらのエラーの発生により，ユーザに対して期待通りのサービスが提供できなくなる可能性がある．そのため，開発者は自身らが公開する Web リソースに対して，エラーの発生状況を継続的かつ正確に把握する必要がある．加え，ユーザ側の実行環境は多種多様であり，ブラウザの種類やバージョン等によってリソースの振る舞いが異なることもある．そのため，開発者はブラウザ互換性の確認に多大な労力を費やす必要がある．

本研究では，公開中の Web リソースに対するエラー発生状況把握の容易化を目的として，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エラーリポジトリを提案する．参加型センシングとは，ユーザの自発的な参加の元に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．エラーリポジトリでは，まずユーザの普段利用するブラウザにプラグインをインストールしてもらう．このプラグインはエラーの発生を検知し，そのエラーに関連する情報を収集サーバに送信する．収集サーバでは，様々なクライアントから送信されたエラー情報を一元管理し公開する．エラー収集をユーザの協力により実施することで，多様な環境からエラーについての情報を収集することができる．また，Web リソース開発時だけでなく継続的にリソースでのエラーの発生状況を把握できる．本論文では，プライバシーを考慮した収集項目の検討，およびエラーリポジトリの実装と公開を行う．評価実験として，複数人のユーザの協力による一定期間のエラー収集を実施する．収集したエラーに関する情報を元にエラーリポジトリの評価を行い，エラー発生状況の把握やエラーの原因解明に役立つ可能性があることを示す．

主な用語

Web フロントエンド開発，エラー収集，参加型センシング

目次

図目次

表 目 次

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 [1]．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 [2]．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 [3]．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 [4]．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 [5]．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 [6]．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[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

[1] https://www.stackoverflow.com

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 [7][8]．提案システムはこの考えを取り入れたエラー情報収集機能とエラー修正例収集機能を持つ．まずエラー情報収集機能では，様々なユーザに自主的に協力してもらい，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．このエラー情報収集機能は先に述べた 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．エラー修正例収集機能では，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．

本論文では特にエラー情報収集機能に着目する．はじめに，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を述べ，予備調査を実施する．続いて，提案システムを構成するブラウザ拡張プラグインとサーバプログラムを実装する．さらに，実装したシステムを用いて約 1 ヶ月間にわたって被験者実験をしたのち，アンケートを実施する．被験者実験により収集したエラーの情報およびアンケートの考察を行う．本論文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
- 被験者実験を行い，エラーの原因解明に役立つ情報が収集される可能性があることを示した

2 準備

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 [9] 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 1 つの Web 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．

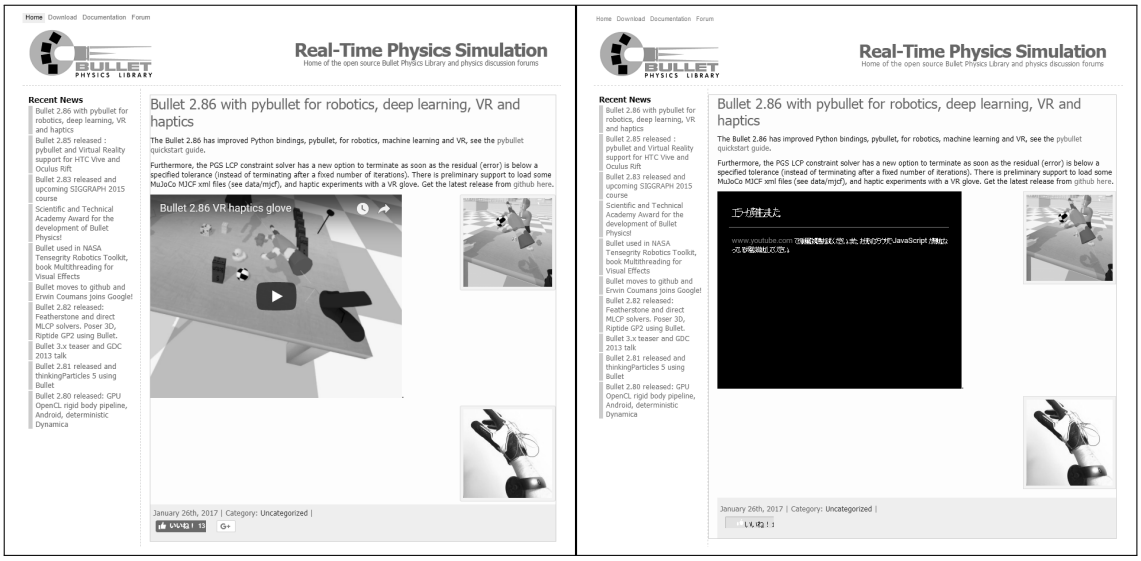
具体的なエラーの事例を示す．図 1 は，ある Web ページ[2]を Google Chrome 61 と Internet Explorer 6 のそれぞれの Web ブラウザで閲覧した画面のスクリーンショットである．IE6 で閲覧した画面の画像を見ると，画面中央の YouTube 動画や画面下部の Facebook のいいねボタンが正しく表示されていない．加え，YouTube 画面の「エラーが発生しました」というエラーメッセージ自体もレイアウトが崩れている．

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 2 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

[2]http://bulletphysics.org/wordpress/ (2017 年 10 月 7 日閲覧)



(a) Google Chrome 61 による Web ページの閲覧　　(b) Internet Explorer 6 による Web ページの閲覧

図 1: Web リソース上で発生するエラーの例

```
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 2: Web ページ内部で発生していたエラー

発者は IE6 利用者を想定して開発し，その後 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 [7]．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より騒音 [10]，大気汚染 [11]，渋滞情報 [12] といったセンシングを行う既存研究が行われている．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 [13]．

また，Pengfei ら [14] はバスの待ち時間に関する研究を行っている．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．GPS の位置情報を利用しなくとも屋内や人ごみにおいて位置を特定する技術も提案されており [15][16]，様々な計測対象がセンシングされている．

一般ユーザが参加型センシングに参加するためにはユーザにとっての利益・インセンティブが重要な要素となる [17]．スマートフォンでセンシングすることにより参加型センシングに協力するユーザは電池消費やプライバシー情報の侵害の危険性が存在するためであるとされている．Juong-Sik らは参加型センシングにおける経済モデルを提案し，分配と社会福祉の公平性を改善したとしている [18]．Irene らは大気汚染のセンシングにゲーミフィケーションを取り入れることで，ユーザを巻き込みやすくなる工夫を施した研究を行っている [19]．

本研究では参加型センシングのコンセプトを取り入れる．参加型センシングの取り組みに協力するユーザを一般的なユーザと区別するために協力ユーザと呼ぶことにする．

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，エラーの早期発見および修正を目的としたシステム，エラーリポジトリを提案する．このシステムは，エラーの原因解明に役立つ情報やエラーの修正例を収集して一元管理し，Web フロントエンド開発者に向けて公開する．公開された情報や修正例によってエラーの修正が容易になれば，開発者のエラーに対する負担の軽減が期待できる．エラーの原因解明に役立つ情報とは，エラーの状況を再現するのに必要な情報である．その情報の中には，エラーメッセージやスタックトレースといったエラーが発生したときに生成される項目や，OS や Web ブラウザの種類・バージョンといったクライアントの環境を表す項目がある．具体的にどの項目で構成されるかについては 4 章の予備調査において述べる．以降，エラーの原因解明に役立つ情報のことをエラー情報と呼ぶことにする．

エラーリポジトリのキーアイデアはエラー情報の収集に参加型センシングの考え方を取り入れる点にある．参加型センシングのコンセプトの下，協力ユーザのクライアント環境において発生した Web リソース上のエラーについて，エラー情報を収集する．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必要がある．Web 全体を大きな物理的空間として見なしたときに，URI が地点であり，Web ブラウザとプラグインがセンサデバイスであり，エラー情報が計測対象である．

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，Web ブラウザ拡張プラグイン（以降，プラグイン）と収集・公開サーバを作成する．

1. クライアント環境で発生したエラー情報収集機能
2. エラー解決時の差分を基にしたエラー修正例収集機能

以降では，前者をエラー情報収集機能，後者をエラー修正例収集機能と呼ぶことにする．それぞれの機能について説明する．

エラー情報収集機能の動作の流れを図 3 に表す．エラー情報収集機能は協力ユーザを必要とする．協力ユーザは普段使用している Web ブラウザにプラグインをインストールする．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，協力ユーザは自身の環境で Web リソースを実行する．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 リソースを協力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，プラグインがこれを検知し，収集・公開サーバに対してエラー情報を送信する（図中矢印 3）．収集・公開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 Web サービスとして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ク

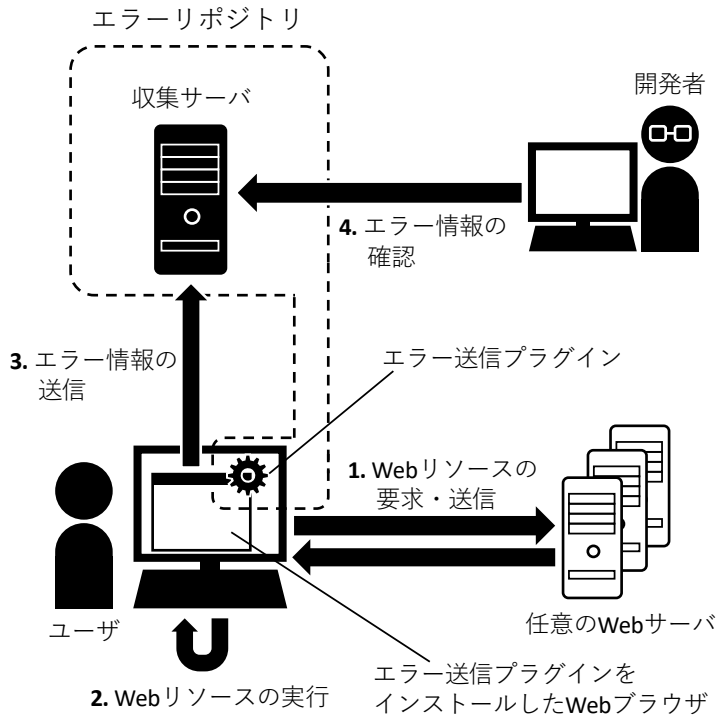


図 3: エラー情報収集機能の仕組み

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

図 4: `window.onerror` の例

ライアント環境におけるエラーの発生状況が把握できるようになる．開発者はエラー情報を基にエラーが発生した状況を再現して，場合によってはエラーの原因を突き止めることができる．エラー情報の具体的な公開方法については，5 章の実装にて述べる．

エラー修正例収集機能では，エラー解決時の差分を基に修正事例を収集する．エラー情報収集機能において協力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が修正に寄与した修正例であることを期待して収集する．収集した修正例はすべての Web フロントエンド開発者に公開される．開発者は修正方法を知りたいエラーについてエラーリポジトリで検索すると類似したエラーにおけるいくつか修正例を修正方法として提案する．

エラーリポジトリの機能のうち，本論文で実施する範囲は，エラー情報収集機能の予備調査から実験，評価までである．エラー修正例収集機能については，今後の展望として 8 章にて述べる．

3.3 エラー情報の収集方法

エラー情報を収集するためシステムを提案するにあたって，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できる．`window.onerror` は図 4 のように独自の処理を実装することができる．そして実行時エラーが発生したときにそれらの処理が呼び出される．例えば，図の実装だと実行時エラーが発生するたびにエラーメッセージがログとして出力される．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．`window.onerror` の中でエラー情報を非同期通信 Ajax を用いて収集・公開サーバに送信する処理を記述することを考える．この方法によりプラグインが Web リソース上で発生した実行時エラーを検知して，収集・公開サーバに自動でエラー情報を送信する仕組みが実現できる．

注意すべき点として，この `window.onerror` により収集できないエラーが存在するという点である．`window.onerror` は JavaScript や HTML を Web ブラウザで実行するときに，エラーが発生すると呼び出される．ゆえに，Web サーバにおいて Web リソースが見つからないというようなエラーについては検知できない．今後そのようなエラーについてイベントハンドラが用意されるかどうかは今のところ不明である．しかしながら，もし用意されれば，本研究で収集できるエラーの範囲が拡張できるといえる．

4 予備調査

4.1 埋め込みエラーによるエラー収集の確認

4.1.1 調査目的

エラー情報の収集方法として error イベントを利用することを 3.3 節にて述べた．本調査では，様々なエラーの種類のそれぞれにおいて，その種類のエラーの原因となる誤りを含む Web ページを閲覧した際に，error イベントが生じるかどうかを確認する．同時に，Web ブラウザ間の差異が原因で発生するブラウザ互換エラーについて発生する可能性を確認する．

4.1.2 調査方法

調査方法は，それぞれの種類のエラーが発生する状況を実際に作成して error イベントの有無を確認するというものである．まず，実際にエラーが発生するコードを埋め込んだ Web リソースを実装する．それらの Web リソースを Web ブラウザによりアクセスして，開発者ツールのコンソールにおいて，error イベントが生じたかどうかを確認する．開発者ツールのコンソールとは Web ブラウザで Web リソースを実行したときに標準出力としてログ等が確認できる仕組みであり，Google Chrome 等の主要な Web ブラウザでは実装されている．error イベントが生じたときはコンソールにその旨が表示される．

発生させるエラーの種類は，一般的なエラーとブラウザ互換が原因で起きるエラーとする．一般的なエラーのサンプルとして，参照エラーや構文エラーといったエラーを埋め込んだ Web リソースを実装した．ブラウザ互換のエラーのサンプルとして，ほとんどの Web ブラウザで対応しているが，ある Web ブラウザでは対応していない API を埋め込んだ Web リソースを実装した．ある Web ブラウザが対応していない API については，”Can I use”という Web サービスの検索結果[3]を参考に選択した．

4.1.3 調査結果

まず，一般的なエラーについて，error イベントの発生を確認できた．例えば，構文エラーを埋め込んだ Web リソースにおいては SyntaxError というエラーが，参照エラーを埋め込んだ Web リソースにおいては ReferenceError というエラーが原因となって error イベントが生じていた．

次に，ブラウザ互換が原因で起きるエラーについても error イベントの発生が確認できた．例えば，Payment Request API というクレジット支払い関係の API は Google Chrome や Edge は対応しているが，Firefox では対応していない．この Payment Request API のコードを埋め込んだ Web リソースにおいて Firefox で error イベントが発生することを確認した．

[3]https://caniuse.com/#compare=edge+16,firefox+56,chrome+62

4.2 エラー情報に含まれるプライバシーに関する項目の調査

4.2.1 調査目的

エラーリポジトリではエラー情報を収集および公開する．そこでまずエラーが発生したときの状況を表す項目一覧の中から収集する項目を決定しなければならない．また，協力ユーザから収集したエラー情報を公開する上で，プライバシーに関する項目が公開されるのは望ましい状態ではない．エラーの原因解明のためにはエラーが発生したときの状況を可能な限り細かく収集する方法が考えられるが，中にはプライバシーに関する内容を含む項目が存在する．項目を網羅するということとプライバシーを守るということはトレードオフの関係にあり，適度なバランスを目指す必要がある．

この予備調査の目的は，エラー情報の項目のうち，どの項目にプライバシーに関する内容が含まれる可能性があるかを調査することである．エラーの発生状況を表す各項目に対して，収集するかどうかの判断に

1. エラーの原因解明に役立つこと
2. ユーザの許可なしに取得が可能（オプトアウト）であること

の 2 点を基準として設ける．次に，収集する項目からなるエラー情報を実際に収集して確認することで，各項目に対してプライバシーに関する内容を含む可能性があるかどうかを調査する．

表 1: エラー発生時の状況を表す項目一覧

項目[4]	例	原因解明に役立つか	オプトアウトか
URL（クエリ部分以外）	http://www.example.com/main.html	✓	✓
URL（クエリ部分）	?id=yamamoto&keyword=apple	✓	✓
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	×	×

[4]濃い網掛けの項目は，5 章の方針決定で収集かつ公開することになる項目である．同様に，薄い網掛けの項目は公開のみすることになる項目である．

4.2.2 調査方法

プライバシーに関する項目の調査の方法を述べる前に，まずエラーの発生状況を表す各項目に対して，エラーの原因解明に役立つかどうかという観点でその項目を評価する．

エラー発生時の状況を表す項目を表 1 にまとめる．なお，エラーの原因解明に全く関係のない項目については一部表示していない．また，後の調査結果の影響により URL がクエリ部分とそれ以外を別の項目としている．各項目とその一例を左側の 2 列に挙げている．これらの項目に対し，その内容を基にエラーの原因が解明できるシナリオを考えることができれば役立つ項目とする．原因解明に役立つかどうか判断した結果は表 1 の右から 3 番目の列に示す．

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く，役立つとは考えられない．それ以外の表中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 はログインした 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．

エラーの原因解明に役立つ項目のそれぞれについて，ユーザの許可なしに取得が可能かどうかという観点でエラーリポジトリとして収集するかどうかを判断する．具体的な基準としてオプトアウトであるかどうかを基準とする．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．すなわち基準によりエラーの原因解明に役立つ項目からオプトアウトでない項目を除外することを意味する．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．オプトインの項目が Web ブラウザ上で取得されようとすると，Web ブラウザはポップアップを表示して許可するかどうかをユーザに確認する．これがエラー情報送信のたびに表示すると協力ユーザの負担が大きくなってしまう．よってオプトアウトでなくオプトインである項目は収集しない．エラーの原因解明に役立つ項目のうち，オプトアウトでなくオプトインである項目は位置情報（GPS）である．そのため位置情報（GPS）は収集しない．以上より，収集する項目は表中の網掛けされている URL，エラーメッセージ，スタックトレース，ブラウザ・OS，日時，cookie とする．なお，濃い網掛けがされているのはエラーリポジトリの実装方針において収集かつ公開するとした項目であり，薄い網掛けがされているのは収集のみ行うとした項目である．詳しくは 5 章で述べる．

エラー情報の項目のうち，プライバシーに関する内容を含む可能性のある項目の調査方法について述べる．この予備調査ではエラーリポジトリのプロトタイプを実装して，複数人の協力ユーザにより一定期間の間プラグインを試用してもらう．期間中に集まったエラー情報を目視で確認し，プライバシーに関する内容を含む項目があるかどうかを調べる．予備実験の段階ではプライバシーに関する

内容が含まれるかどうかを調べることを目的としているため，プロトタイプにおける収集・公開サーバでは収集のみを行い，エラー情報の公開はしない．

プラグインは JavaScript，収集・公開サーバは Node.js （サーバサイド用の JavaScript の実行環境）で実装した．協力ユーザの内訳は Google Chrome の使用者が 7 名，Firefox の使用者が 1 名である．予備実験のエラー情報収集は 2017 年 8 月 15 日から 2017 年 8 月 31 日までの約 2 週間実施した．

4.2.3 調査結果

プロトタイプの 2 週間の運用の結果，4172 件のエラー情報が収集された．ただ，実際は重複したエラー情報が多く，日時以外の項目でユニークなエラー情報は約 500 件であった．これらのエラーについて項目の内容を目視で確認した．

目視の結果，以下のような形でそれぞれの項目にプライバシーに関する内容を含む場合があることが分かった．

- URL のクエリ部分にユーザ ID・キーワード
- URL のパス部分やエラーメッセージに識別 ID
- cookie にセッション ID

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．この含有は多くのエラー情報においてみられた．

```
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識別 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．この含有は Google ドライブのサービスで発生したエラーのみで発見された．

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

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

cookie にセッション ID が含まれることがあった．セッション ID を公開するとその Web サービスにおいてログイン乗っ取り等が行われてしまう危険性がある．この含有は多くのエラー情報で発見された．

```
session_id=xxxxxxxxx
```

5 実装

5.1 予備調査に基づくエラー情報の収集・公開方針の検討

この節では，予備調査において調べた，プライバシーに関する内容を含む可能性のある項目に対する方針といった具体的な収集方針の検討を行う．

はじめに，Cookie と URL のクエリ部分についての収集・公開について検討する．これらの項目においてプライバシーに関する内容が含まれている状況は多くのページで確認された．Cookie や URL のクエリ部分はデータを key と value の組み合わせで保存する．これらのうち key がセッション ID であったりアカウント ID であったりするとプライバシーに関する内容であることがある．ただ，これらの key の書き方は統一されていないので機械的にどれがプライバシーに関する内容なのか判断することは難しい．そのため，デフォルトでは Cookie と URL のクエリ部分は収集しないことにする．ただ，プラグインにおいて収集を許可する設定ができるように実装する．貢献に意欲的な協力ユーザに対してオプトインで収集できるようにしておく．収集した場合でも公開はしない方針をとる．収集した Cookie や URL のクエリ部分については，本論文では実現していないが，エラーが発生した Web リソースの開発者のみに対して譲渡する仕組みを考えている．これについては今後の展望として，8.3 節で述べる．また，エラー情報を公開する Web ページにおいてはそこに何らかの Cookie やクエリが存在したことを伝えるために `http://www.example.com?[query]` のように Cookie やクエリの部分の文字列を `[cookie]` と `[query]` にマスキングして公開する．

次に，URL のパス部分とエラーメッセージについての収集・公開について検討する．これらの項目には，Google ドライブという Web サービスのページのみにおいて，プライバシーに関する内容 (具体的には特定のフォルダに対応した識別 ID) が含まれた．他の Web サービスで同様の含まれ方をする可能性は否定できない．そういった内容がエラー情報として公開され続けるという状況は望ましくない．しかしながら，URL のパス部分やエラーメッセージにおいてどの部分にプライバシーに関する内容が含まれているか判断が難しい．そのため，協力ユーザに削除の判断を委ねることにする．自身のクライアント環境から送信されたエラー情報一覧を確認できるようにし，その中に公開されたくないエラー情報があれば，削除を依頼できるようにする．それ以外の対処法としては，クライアント環境で協力ユーザの設定によりマスキングするか，サーバ側でルールを定めてマスキングして公開するなどの方法があるが現状では一番容易な手段を取ることとした．

予備調査で収集したエラー情報を集計した際，ユニークなエラー情報は全体の 8 分の 1 程度であった．すなわち，同じページの同じエラーが複数回送信されていた．具体的な例として，画面をスクロールするときにエラーが発生する Web ページにおいて，そのエラー情報が短時間の間に大量に送信されるという場合があった．これらの情報を全て送信すると，協力ユーザのネットワーク通信量が増加してしまう可能性がある．また，送信済みのエラー情報を確認できるように保存する上でストレージを逼迫する可能性がある．さらに，送信済みのエラー情報を確認するときにそれらのエラーがノイズとなって，自身のクライアント環境から送信されたエラー情報の把握が困難になる可能性があ

る．そこで，同じページの同じエラーについては，プラグインがインストールされたあるクライアント環境において 1 日に 1 回だけ送信するようにする．エラー情報を送信する期間の最低を 1 日とした理由は，Web リソースの変更によってエラーの発生状況が変化するのを確認する期間として日単位で考えるのが良いと判断したためである．

5.2 処理の流れ

エラー情報収集機能の処理の流れは

- クライアント環境からエラー情報を収集する
- 開発者がエラー情報を検索・閲覧する

の 2 つのユースケースに大別できる．前者をエラー情報収集時の流れとして，後者をエラー情報公開時の流れとして説明する．それぞれのユースケースごとに，どのように実装したのかを理由とともに説明する．実装したエラーリポジトリが動作している画面は付録において記載する．

5.2.1 エラー情報収集時の流れ

クライアント環境からエラー情報を収集するときの流れを説明する．収集関係の処理をさらに分割すると，収集項目の設定，エラー情報の送信，エラー情報の確認および削除の 3 点となる．

収集項目の設定

収集する前の段階として，ユーザが収集項目の設定ができるようにする．具体的には，予備調査でプライバシーに関する内容を含む可能性があるとした項目のうち，URL のクエリ部分と Cookie について収集するか否かを設定できるようにする．デフォルトでは収集しない設定となっている．協力ユーザが収集する設定にした場合にのみ収集を行う．設定画面のイメージは付録の図 5 のような画面である．それぞれの項目についてチェックボックスをクリックして保存すると収集する設定となる．この設定内容は Web ブラウザがプラグイン用に提供するストレージに保存する．あとで説明するエラー情報送信時に，収集項目の設定の確認を行う．収集しない設定になっている場合は，それらの項目の内容をマスキングしたのち送信する．

エラー情報の送信

プラグインによりエラー情報を収集・公開サーバに送信する．説明が前後するが，プラグインのストレージには過去に送信したエラー情報一覧のリストが保存されている．送信時にその日に送信したエラー情報一覧のリストを確認する．もし，その中に URL とエラーメッセージが同じエラー情報がなければ，エラー情報を収集・公開サーバに送信する．ストレージに保存されたデータのうち，その日に送信したエラー情報のリストだけを参照・逐次探索するので，長期間にわたって使用するほど重複確認の処理が長くなるといった問題は起きない．エラー情報

の各項目は window.onerror メソッドに引数で与えられる error オブジェクトや，ブラウザ・OS といったクライアント環境を表す navigator.userAgent オブジェクトから取得する．送信は API の形で非同期通信で行われ，JSON 形式でエラー情報を表現する．通信は HTTPS プロトコルにより行われるので，エラー情報が通信経路において他者が容易に中身を見ることはない．エラー情報を受信する収集・公開サーバはエラー情報をサーバ上のデータベースに保存する．送信と同時にプラグインのストレージにもエラー情報を保存する．エラー情報の集合は日付ごとにまとめてリストに保存する．

エラー情報の確認

協力ユーザは自身の環境から送信したエラー情報を一覧を確認できる．Web ブラウザのツールバーにエラーリポジトリのアイコンを表示させる．そのアイコンをクリックすると送信済みエラー情報の一覧を確認することができる．送信済みエラー情報の一覧はプラグインのストレージに保存されたデータを基に表で表示される．送信済みエラー情報閲覧画面のイメージは付録の図 6 のような画面である．ユーザが一覧を確認したときに，公開して欲しくないエラー情報が含まれている場合はエラー情報の隣にある削除依頼ボタンを押下することによってプラグインのストレージに保存されたエラー情報だけでなく，収集・公開サーバのデータベースに保存されたエラー情報も削除する．エラー情報の保存方法においては匿名性のためにエラー情報ごとの ID は付与していない．そのため，エラー情報の削除は（URL，日時，ユーザエージェント）の組み合わせで収集・公開サーバに API を発行する．収集・公開サーバでは（URL，日時，ユーザエージェント）の組み合わせでエラー情報を検索し，もし該当するエラー情報があれば，1 件削除する．万が一同じ時刻に同じ環境で同じエラーが発生したとすると別のエラーが削除される可能性もあるが，エラーの原因解明をする開発者にとってはほぼ同じ価値であるために，このような方法を取った．プラグインのストレージに保存された送信済みのエラー情報のデータは協力ユーザの必要に応じてクリアすることができる．

5.2.2 エラー情報公開時の流れ

Web フロントエンド開発者が自身が開発するドメインの Web リソース上で起きているエラー情報を確認する流れについて説明する．収集・公開サーバは Web サービスの形でエラー情報を公開する．この Web サービスのページを以降では公開ページと呼ぶ．公開ページのインデックスページの画面イメージは付録の図 7 のような画面である．2018 年 2 月 7 日現在，この公開ページには https://tyr.ics.es.osaka-u.ac.jp/erepo/ の URL からアクセスすることができる．インデックスページの冒頭部には検索窓があり，URL によってエラー情報を検索することができる．また，エラーリポジトリの概要や，付録の図 8 のように最近発生したエラーの情報や発生したエラーのカテゴリの分布が確認できる．

開発者は自身のドメインでエラー情報を検索する．インデックスページにおいて入力したキーワー

ドに URL が部分一致するエラー情報の一覧を表示する．検索結果の画面イメージは付録の図 9 のような画面である．開発者はエラー情報の集合を 10 件ずつページングで閲覧することができる．また，直近の 25 日間の間のエラーの発生状況をグラフで確認することができる．一覧のそれぞれのエラー情報に対して表の一番右の列にあるリンクをクリックするとエラー情報の詳細画面に遷移する．詳細画面の画面イメージは付録の図 10 のような画面である．収集・公開方針で述べたエラー情報のそれぞれの項目が確認できるようになっている．開発者はこれらの情報と自身が開発する Web リソースを照らし合わせることでエラーの原因解明および修正に役立てることができると考えられる．

5.3 実装に用いたツールと成果物

プラグインは 3.2 節で述べたように，ユーザの Web ブラウザにインストールされ，エラー発生時にエラー情報を収集・公開サーバに送信する．プラグインは JavaScript で実装する．Google Chrome 用と Firefox 用のプラグインを実装する．ソースコードは GitHub にて公開している[5]．

収集・公開サーバはプラグインから受信したエラー情報を保存する．保存しているエラー情報を集計等を行なった形で公開する．収集・公開サーバは主に Java で実装する．また，フレームワークとして Spring Boot を利用して開発を行う．ビューの部分は Bootstrap を利用して実装する．グラフの描画には D3.js ライブラリを利用する．ソースコードは GitHub にて公開している[6]．

[5]https://github.com/myamamt/erepo-plugin
[6]https://github.com/myamamt/erepo

6 実験

6.1 実験の目的

この実験では，協力ユーザによるデータ収集を行う被験者実験を実施する．実験を実施する目的は 2 つある．1 つ目の目的は，実際に協力ユーザにプラグインを試用してもらうことで自動でエラー情報を集めることにより，クライアント環境ではどういったエラーが起きるのかを確認することである．もう 1 つの目的は，プラグインおよび収集・公開サーバを実際に運用することで試用中に感じたことを協力ユーザから聞き取ることである．参加型センシングは，協力ユーザが多いほど観測データがリアルタイムかつ正確になる．協力ユーザからの意見を今後のツールに反映させることでユーザビリティが向上し，参加しやすいシステムを目指す．また，今回は協力ユーザの視点だけでなく，エラーリポジトリを利用する開発者の視点での質問も追加してエラーリポジトリの公開方法について客観的な評価を行う．

6.2 実験内容

6.2.1 被験者によるデータ収集

実験の内容は，複数人の協力ユーザによる 1 ヶ月間の試用と使用後に実施するアンケートである．協力ユーザは，研究室に所属する教員および学生の 15 名である．

各協力ユーザには，普段試用している Web ブラウザの種類を回答してもらい，その Web ブラウザに対応したエラー送信プラグインをインストールしてもらった．そして，ブラウジング中にエラー情報を収集する．1 ヶ月間の試用の後，試用中に感じたことや本研究に対する意見等をアンケート形式で回答してもらった．アンケートはリッカート尺度 [20][21] を使用して作成する．

実施期間は 2017 年 12 月 21 日から 2018 年 1 月 25 日までの約 1 ヶ月間である．協力ユーザの使用 Web ブラウザの内訳は，Google Chrome13 名，Firefox1 名，Vivaldi1 名である．また，実施したアンケートの内容は付録の表 4 に記載する．

6.2.2 クローリングによるデータ収集

被験者によるデータ収集とは別に，クローリングによるデータ収集も行う．被験者によるデータ収集では，収集できるエラー情報の母数に限りがある．そこで，ある協力ユーザが特定の分野の Web サイトにアクセスしたと仮定して，クローリングによるデータ収集を行う．これにより，エラー情報の母数の増加，および特定の分野の Web サイトに限定したエラーの収集を目的としている．

まず，協力ユーザを想定したクライアント環境を用意する．すなわち，あるマシンの Web ブラウザにエラー収集プラグインをインストールする．その Web ブラウザに対して，新たに実装したアクセス用プラグインをインストールする．このアクセス用プラグインは指定した Web サイトのリストに対して逐一アクセスしていくプラグインである．アクセスの際は，Web リソースの読み込みが完了

してから 10 秒間待機してタブを閉じる．これは，読み込みが完了してから数秒間の間に JavaScript の処理が実行される可能性を考慮した設計になっている．アクセスしてエラーが起きるとエラー収集プラグインによりエラー情報が送信されるため，リスト上の Web サイトにおけるエラー情報が収集できるという仕組みである．なお，クローリングを実施する環境は Windows10，Chrome63.0 である．

クローリングする分野については，社会に影響力のある分野をいくつか選択する．クローリングする分野を表 2 にまとめる．

さらに，クローリングした分野のうち，コンピュータサイエンス関連の国際会議において，エラーリポジトリの評価方法の 1 つとして開発者への報告を行った．その内容は，実験で観測されたエラーについてその会議を運営している人々に報告をするというものである．収集したエラーについて公開した情報を開発者に実際に見てもらうことでエラーの修正に役に立ったかどうかを判断してもらう．

この実験に対して，エラーリポジトリの初期段階の設計としてユーザに協力を求めるのでなく，クローリングのみで実施すれば良いのではないかという疑問に対してあらかじめ回答しておく．このクローリング実験では，特定の分野の Web サイトのエラー情報を収集することができるが，多種多様な環境におけるエラー収集は実現できていない．また，Web サイトにアクセスしてからクリック等のアクションを何もしないままにタブを閉じるので，アクションにより発生するエラー情報につい

表 2: クローリングする分野

分野	リストの作成方法	サイズ	クローリング日時
コンピュータサイエンスに関する国際会議	WikiCFP[7]という国際会議をまとめた Wiki サービスを利用する．2017 年 12 月 31 日時点でコンピュータサイエンスカテゴリの 20 ページ目までに載っている約 400 会議の URL をスクレイピングによりリスト化して作成	377 件	2017 年 12 月 31 日
東証上場企業	日本取引所グループが公開しているリスト[8] から上場企業名の一覧を取得し，”I’m Feeling Lucky”という Google 検索の機能と組み合わせてリストを作成	3897 件	2018 年 1 月 4 日
トラフィック数上位のサイト	Alexa という企業が公開している世界中のトラフィック上位のサイトのリスト[9] を編集して作成	500 件	2018 年 1 月 11 日

[7]http://www.wikicfp.com/cfp/
[8]http://www.jpx.co.jp/markets/statistics-equities/misc/01.html
[9]https://www.alexa.com/topsites

ては収集できない．エラーリポジトリでは，多くの Web サイトからエラー情報を収集できるという利点の他に多種多様な環境およびアクションからエラー情報を収集できるという利点がある．そのため，クローリングだけではエラーリポジトリの目的は達成できない．

6.3 実験結果

エラー情報の総数および収集したエラー情報をエラーの種類ごとに分類してそれぞれの数をカウントしてまとめたものが表 3 である．なお，クローリングによるデータ収集の列のうち，「国際会議」の列はコンピュータサイエンスに関連した国際会議のサイトのクローリング，「上場企業」の列は東証上場企業のサイトのクローリング，「TF 上位」はトラフィック数上位のサイトのクローリングそれぞれの実験結果を表す．

実験期間の間に被験者によるデータ収集で集まったエラー情報の総数は 873 件であった．これは，1 日あたり 25 件のエラー情報が被験者のクライアント環境から送信されていたことになる．エラーの種類については，参照関係のエラーである ReferenceError が最も多く（56.0%），次いで型関係のエラーである TypeError が多かった（26.3%）．

アンケートの結果は付録に記載している．選択式の 4 問の質問についてはそれぞれの分布を円グラフで表している．自由記述については要約したものを箇条書きしている．

表 3: 収集したエラー情報のエラーの種類

エラーの種類	説明	被験者による	クローリングによるデータ収集		
		データ収集	国際会議	上場企業	**TF** 上位
ReferenceError	未定義の変数を参照したとき等に発生する	489（56.0%）	22（28.6%）	102（28.6%）	16（57.1%）
TypeError	関数でない変数を関数呼び出ししたとき等に発生する	230（26.3%）	41（53.2%）	197（55.2%）	8（28.6%）
SyntaxError	処理が構文を満たさないときに発生する	91（10.4%）	11（14.3%）	49（13.7%）	3（10.7%）
Error	分類できないエラーを呼び出すとき等に発生する	38（4.4%）	2（2.6%）	2（0.6%）	0（0%）
SecurityError	異なるプロトコルの通信等が原因で発生する	16（1.8%）	0（0%）	1（0.3%）	1（3.6%）
その他	それ以外のエラー	9（1.0%）	1（1.3%）	6（1.7%）	0（0%）
計		873	77	357	28

クローリングによるデータ収集でも合計約 450 件のエラー情報を収集した．エラーの種類の多かったエラーについては，トラフィック上位のサイトのクローリングでは ReferenceError と TypeError の大小の順番が同じである一方，国際会議のサイトや東証上場企業のサイトのクローリングでは順番が逆転する結果となった．

また，国際会議のサイトから収集したエラーのうち 10 件を選択してそれぞれの国際会議のコンタクト先のメールアドレスに対してエラーが発生していることとそのエラー情報を報告した．これに対し，ある国際会議の運営者から，エラーの報告への感謝および修正したという旨のメールが届いた．

7 考察

7.1 集計結果の考察

被験者から収集したエラー情報の総数は 873 件であった．これは 1 日あたり約 25 件のエラーが被験者から収集されたことになる．通常のブラウジングにおいて，日常的にエラーが発生していることがわかる．

エラーの種類の割合を見てみると，被験者によるデータ収集およびクローリングによるデータ収集のどちらにおいても大半が ReferenceError，TypeError，SyntaxError の 3 種類のエラーが占めていることがわかる．被験者によるデータ収集およびトラフィック数上位のサイトのクローリングでは ReferenceError が最も多いのに対し，国際会議や上場企業のクローリングでは TypeError が多い．母数を考慮せずに値のみから判断するならば，協力ユーザによるエラー情報収集では国際会議や上場企業のサイトというよりはトラフィック数上位のサイトから収集することが多い可能性があるといえる．

クローリングによるデータ収集において，3 種類のクローリング結果を比較すると，1 サイトあたりのエラー数に差があることがわかる．1 サイトあたりのエラー数はエラーの発生数をリストのサイズで割ることで求める．国際会議のサイトのクローリングでは 0.20 件/サイト，東証上場企業のサイトのクローリングでは 0.09 件/サイト，トラフィック数上位のサイトのクローリングでは 0.06 件/サイトであった．トラフィック数上位のサイトは規模の大きい企業が運営していることが多く，Web リソースの開発に充てられる予算が多いため，エラーが発生する割合が低いのではないかといった仮説が考えられる．一方，国際会議のサイトは Web フロントエンド開発の専門家がページを開発しているわけではないので，エラーが発生する割合が比較的高くなっているのではないかと考えられる．

7.2 収集したエラーの事例

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．開発時だけにエラーの有無を調べるのでなく，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エラーの原因が判明した事例を紹介する．

```
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[10]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>
```

[10]https://icsme2017.github.io/

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[11]において，以下のようなソースコードも発見できた．

```
} 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．

7.3 アンケートの回答による評価

アンケートの回答について，表 4 の上の質問から順に考察していく．

「プラグインのインストール後，Web ページの表示にかかる時間が長くなりましたか？」の質問については，ほどんどの被験者が「全くそう思わない」と回答した．この結果は，プラグインのインストールによって協力ユーザの Web ページ閲覧が影響を受けることはほぼないといえる．

「プラグインの送信済みエラー情報閲覧画面を何回確認しましたか？」の質問については，「4〜10 回」「1〜3 回」「0 回」がそれぞれ 7 人，4 人，4 人という結果になった．「4〜10 回」と回答した被験者は 3 日から 1 週間に 1 度閲覧画面を確認していることになり，自身のクライアント環境から送信されたエラー情報に関心があると考えられる．一方「0 回」と回答した 4 人のようにどういったエラーが発生しているのかについてあまり興味がないと思われるユーザもいることがわかった．

「エラー情報の中に，アカウント ID を含む URL のような，個人が限定できる項目が存在することがあります．存在することに抵抗を感じますか？」の質問については「全くそう思わない」または「そう思わない」と答えた被験者が 8 人いる一方,「そう思う」または「強くそう思う」と答えた被験者が 5 人いた．比較的抵抗を感じない被験者が半数を超えていたということで，インセンティブを適切に付与できれば，一定数のユーザが参加が見込める可能性がある．

先の質問で「そう思う」または「強くそう思う」と回答した被験者に対して実施した「具体的にどのような点に抵抗を感じますか？ 」という自由回答形式の質問への回答には，SNS 等の趣味に使うアカウントが特定される点という回答があった．この回答は，あるアカウントがエラーリポジトリに参加していることが判明することに抵抗を覚えている，またはアカウント ID と同時に他の項目が収集されることによりアカウントが限定されることに抵抗を感じているものと考えられる．このようなユーザの参加も期待できるようにする方法として，エラー情報に含まれるアカウント ID を検知して

[11] https://mj-news.net 2017 年 10 月 7 日閲覧

マスキングして公開する手法の提案が望まれる．また，URL を知っていれば誰でも閲覧できる URL の問題については，実験で発見されたエラー情報の URL では URL に含まれる識別 ID がランダムに生成された文字列であったので，URL のランダムらしき部分をマスキングする方法が考えられる．

「エラー情報収集に協力するユーザにとって，プラグインにあったらいいと思う機能はありますか？」という質問には，協力ユーザのインセンティブに関する回答がいくつか見られた．エラーリポジトリは利益を上げるモデルではないので，金銭的なインセンティブをユーザに提供することは難しい．しかしながら，バッチやオーバレイ表示によりユーザの意欲を向上させる方法は検討する価値のある回答であった．これらの回答から発想した今後の展望について 8 章で述べる．

「実験中に公開ページを何回閲覧しましたか？」という質問については 4 回以上の閲覧した被験者はいなかった．この結果からは被験者は自分以外のユーザから収集されたエラー情報についてはあまり関心がないものと考えられる．

公開ページにあったら良い機能についての質問では，URL でなくエラーメッセージで検索できるようにしてはどうかという回答があった．確かに同じエラーメッセージのエラー情報が検索できれば，共通点からエラーの原因を見出せる可能性がある．また，エラーの発生を開発者に通知する機能や各エラー情報についてデベロッパが議論できる機能が回答でみられた．これらの回答から発想した今後の展望について 8 章で述べる．

7.4 その他

国際会議のサイトの開発者である運営者にエラー情報を報告することで報告を感謝していることとエラーを修正したということの文面の返答があったことについて，これはエラーリポジトリが他者から客観的に評価されたことを表す．今後の方針として，Web フロントエンド開発者に対して自動で積極的に報告することにより，エラーリポジトリが利用されるようになる可能性があるといえる．

ReferenceError の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のような Web リソースをブロック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ときに原因が解明できるエラーであるが，開発者にとってノイズになりうる可能性があるため，今後の収集方針改善において検討する必要がある．

8 今後の展望

8.1 ユーザに対するインセンティブの検討

本論文中では議論しなかったが，協力ユーザがエラーリポジトリに参加する動機付けを考える必要がある．本研究では，Web フロントエンド開発者がエラーリポジトリを利用する際に料金を徴収するといったことはしないので，協力ユーザに対して金銭的なインセンティブを付与することは難しい．

そこで，アンケートの回答から得られたアイデアを基に，達成感を刺激するゲーミフィケーションのコンセプト [22] を取り入れることで協力ユーザに対してインセンティブを提供する方法を考える．

エラー情報を送信する際に楽しませる工夫をする方法がある．例えば，送信時にエラーリポジトリのアイコンが震えたり輝いたりするアニメーションを追加することが考えられる．

協力ユーザのランキングを設置することで複数ユーザ間の競争を期待する方法も考えられる．協力ユーザが希望すればランキングのためだけに使用する ID を割り当てて，エラー情報の送信数でランキングを作成するといったアイデアである．過去に収集されていない新規のエラーであったり，その他原因解明に役立ちそうなエラーに高ポイントが付与されるような基準を設けることができれば，ユーザの意欲を刺激しつつ，質の良いエラー情報が収集される可能性もある．

また，ゲーミフィケーションの観点以外にも親しみやすいデザインに変えていくというのも重要である．デザインが協力ユーザのモチベーションに影響する可能性はある．そのため，Web の進化に合わせて適宜デザインを変更するべきである．

8.2 エラー修正事例の収集機能

本論文ではエラーリポジトリのエラー情報収集機能について実装から考察までを行なったが，提案したもう 1 つの機能であるエラー修正例収集機能について展望を述べる．

エラー修正例収集機能がエラー情報収集機能と大きく違う点は，収集して公開する内容が全ての Web フロントエンド開発者のために公開されることにある．Web 上で行われたエラーの修正らしきコード片を継続的かつ大量に収集して公開することで変化の流れが大きい Web においても常に最新の修正例が提供できることが期待できる．

エラー修正例収集機能の動作の流れは 3 章で述べたとおりである．ただ，この機能の研究をするためには，変更の差分が修正である事例がどれくらい存在するのかを調査して確かめる必要がある．もし，変更の差分が修正である事例が多数見つかるのであれば，変更のどの部分が修正に寄与しているのかを判断する必要がある．JavaScript の抽象構文解析木を用いる既存研究 [23] や段階的情報フローを用いる既存研究 [24] を参考にして差分を分割してエラーが発生しなくなるかどうかを確かめる方法が考えられる．また，定期的に Web リソースを取得する間隔についてもサーバの負荷等を考慮して妥当な間隔を設定する必要がある．

8.3 開発者への通知および情報譲渡

エラー情報を Web フロントエンド開発者が毎回確認しにくるのではなく，メールによって通知できる仕組みになれば，開発者の負担がさらに軽減できると考えられる．しかしながら，エラー情報を通知するためにはエラーが発生したサイトにおける開発者のメールアドレスが分からなければならない．

そこで，各開発者がメールアドレスを指定する方法として Robots.txt の仕組みを参考にした仕組みを考える．Robots.txt とは，指定の場所に robots.txt というテキストファイルを設置することで，Google のクローラがインデックスに登録する方法を制御したり，クロールの最低間隔を定めたりする仕組みである [25]．同様にメールアドレスを記載したテキストファイルをドメイン直下に設置してもらうことで，エラーリポジトリが自動でメールアドレスを取得してエラー情報を送信できるようになる．

現時点のエラーリポジトリでは，エラー情報の中でエラーの原因解明に役立つ可能性のある項目のうち，プライバシーに関する項目については収集はするが，公開はしていない．一般に公開しないが，エラーが起きたページの開発者のみに対して公開する方法として，暗号鍵方式 [26] を用いて譲渡する方法が考えられる．まず，開発者は公開鍵と暗号鍵を生成する．先ほど述べた Robots.txt を模倣した仕組みにおいて，テキストファイルに開発者の公開鍵を記載する．エラーリポジトリはこの公開鍵を取得して，その鍵を使用してエラー情報を暗号化する．暗号化したエラー情報はダウンロードできるようにしておく．開発者はこれをダウンロードして秘密鍵により復号化し，プライバシーに関する項目も含めてエラーの原因解明に活用するといった方法である．

9 関連研究

9.1 既存のエラー収集サービス

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[27] や Airbrake[28] 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.onerror` 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，およびプライバシーに関する内容について考慮する必要があるが，それらは論文中で述べたとおりである．

9.2 Web リソース上のエラーに関する既存研究

Frolin ら [9] は JavaScript のエラーの実情を調査した．調査の中でエラーの種類を分類して，集計している．調査で述べられている種類は PD，NE，US，SE，その他であり，先頭 4 種類は本研究における SecurityError，TypeError，ReferenceError，SyntaxError に対応する．2011 年に発表されたこの論文では数が多い順に PD（52%），US（28%），NE（9%），SE（4%）であった．本研究と PD に対応する SecurityError の割合が大きく異なる結果となった．この数年間の間にクロスドメインに対する Web ブラウザの仕様変更により，割合が大きく変わったものと考えられる．このような調査に対し，エラーリポジトリはエラー情報を継続して収集するため，エラーの種類の割合をリアルタイムに更新することができるという利点がある．

10 妥当性の脅威

収集・公開項目の検討

本研究では，収集および公開する項目をプライバシー等を考慮して選択をした．しかしながら，それぞれの項目について妥当であるかどうかは調査していない．すなわち，どの項目がエラーの原因解明に寄与しやすいかについて客観性のある評価はしておらず，開発者に対して委任している．そのため，決定した収集・公開項目に過不足がある可能性がある．基準を定めてそれぞれの項目を評価する必要があると考えている．

エラー情報の目視確認

本研究では，プライバシーに関する内容を含む項目があるかどうかの確認およびエラーの事例の確認を目視にて実施している．プライバシーの項目の確認については固有名詞や ID に注意して目視をしたが，その方針が不適切であるまたは見落としが生じている可能性がある．今後のエラーリポジトリの展望としてプライバシーに関係したトラブルが起きないようにするためにも，客観的にプライバシーを含むかどうかを判断できる仕組みが必要であると考えている．

被験者実験

本研究では 15 名の被験者の協力により被験者実験を実施した．本研究は被験者が閲覧した Web ページで発生したエラーの情報を収集するため，被験者の傾向により収集されるエラー情報の内容に偏りがある可能性がある．この偏りを軽減するためには，多種多様な背景を持つ被験者に参加してもらうことと被験者の母数を増やすことが必要がある．また，被験者のほとんどが Google Chrome を利用していたため，ブラウザ互換によるエラーが存在するのかを判定することができなかった．多種多様なクライアント環境を利用する被験者で再度実験を行うことで，ブラウザ互換によるエラーも検出できる可能性がある．

11 おわりに

本論文では，Web フロントエンド開発者のための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，提案システムを構成するブラウザ拡張プラグインとサーバプログラムを実装した．実装したシステムを用いて約 1 ヶ月間にわたって被験者実験をしたのち，アンケートを実施した．被験者によるデータ収集では 1 日あたり約 25 件のエラーの情報を収集できた．収集したエラーの情報の目視確認およびアンケート結果の考察により，エラーの原因解明に役立つ情報が収集される可能性があることを示した．

今後の研究課題として，本論文で実装したエラー情報収集機能を拡張するのであれば，収集・公開方針を検討した項目の妥当性の調査，すなわちエラーの状況を表す項目のうち，どの項目がエラーの原因解明に寄与するのかを調査する必要があると考えられる．また，本論文では検討しなかったエラー収集に協力するユーザに対するインセンティブを検討する必要がある．さらに開発者に対してエラーの発生を通知またはエラーの情報の全項目を譲渡することによって，エラーに関する負担を軽減する仕組みを実現することが考えられる．

エラーリポジトリのもう 1 つの機能であるエラー修正例収集機能について，実装・評価を行うのであれば，まずはじめに差分からエラーの修正例を収集できるかどうかを事前に調査する必要があると考えられる．差分から修正例が収集できるようであれば，定期的に Web リソースを取得する間隔やクライアント環境での実行を再現する方法を検討することでエラー修正例収集機能が実現できると考えられる．

謝辞

本研究を行うにあたり，常に励まして頂き，暖かく見守っていただきました 楠本 真二 教授に心より感謝申し上げます．

本研究を行うにあたり，新しい視点からのご意見を頂き，研究室生活の中で相談に乗っていただきました 肥後 芳樹 准教授に心より感謝申し上げます．

本研究の全過程を通し，日頃より熱心なご指導ご鞭撻を頂き，活発な議論に応じていただきました 柗本 真佑 助教に心より感謝申し上げます．

日々の研究室生活において，声をかけて頂き，様々な問題に対して深い理解を前提とした助言を頂きました，卒業生の 小倉 直徒 氏，同 佐飛 祐介 氏，同 鷲見 創一 氏，同 古田 雄基 氏，同 幸 佑亮 氏，同 横山 晴樹 氏に心より感謝申し上げます．

常に様々な相談に乗って頂き，また日々の研究室生活を豊かにして頂きました，大阪大学大学院情報科学研究科コンピュータサイエンス専攻博士前期課程 2 年の 下仲 健斗 氏，同 中島 弘貴 氏，同 山田 悠斗 氏に心より感謝いたします．

日々の研究室生活の中心となり，研究室を盛り上げて頂きました，大阪大学大学院情報科学研究科コンピュータサイエンス専攻博士前期課程 1 年の 有馬 諒 氏，同 佐々木 美和 氏，同 谷門 照斗 氏，同 内藤 圭吾 氏，同 松尾 裕幸 氏，同 山田 涼太 氏に心より感謝いたします．

短い間でしたが，様々な場面において親切なご助力を頂きました，大阪大学基礎工学部情報科学科 4 年の 田中 紘都 氏，同 土居 真之 氏，同 松本 淳之介 氏，同 林 純一 氏に心より感謝いたします．

最後に，本研究に至るまでに，講義やセミナー等でお世話になりました大阪大学大学院情報科学研究科の諸先生方に，この場を借りて心より御礼申し上げます．

付録

エラーリポジトリ画面イメージ

実装したエラーリポジトリの画面イメージを並べる．それぞれの画面イメージは 5. 実装の章の説明時に適宜参照される．

図 5: プラグイン 設定画面

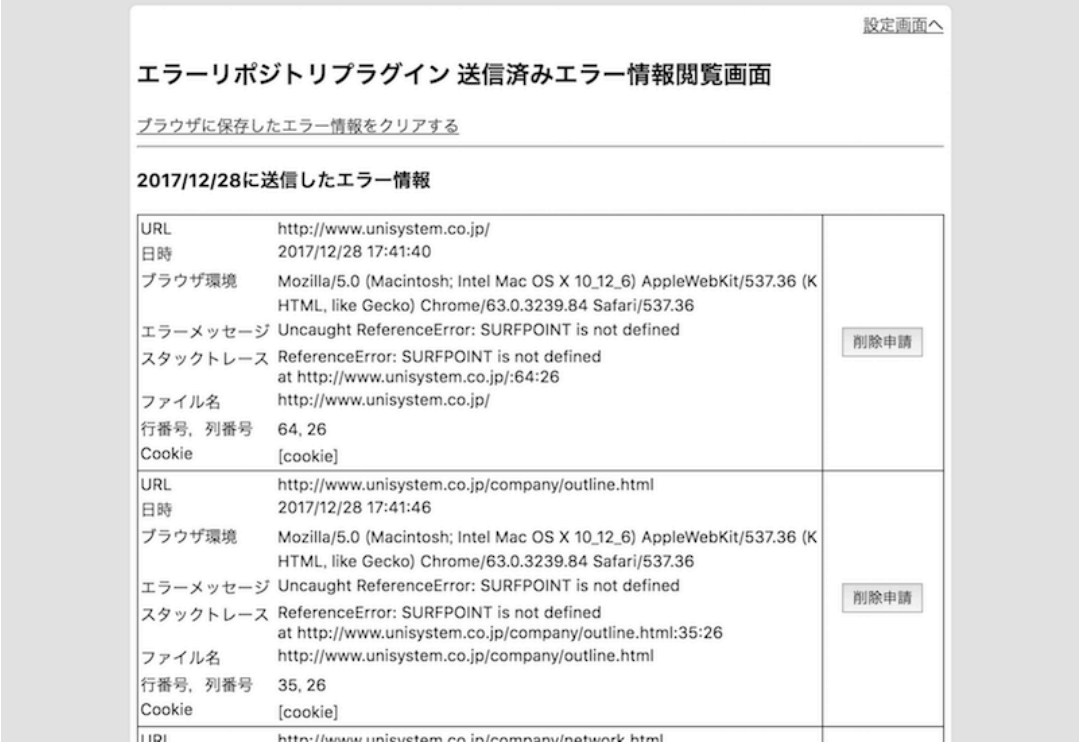


図 6: プラグイン 送信済みエラー情報閲覧画面

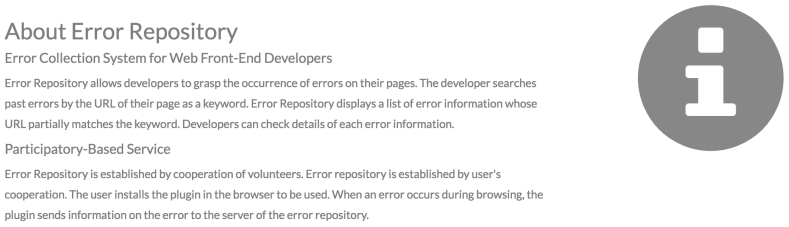
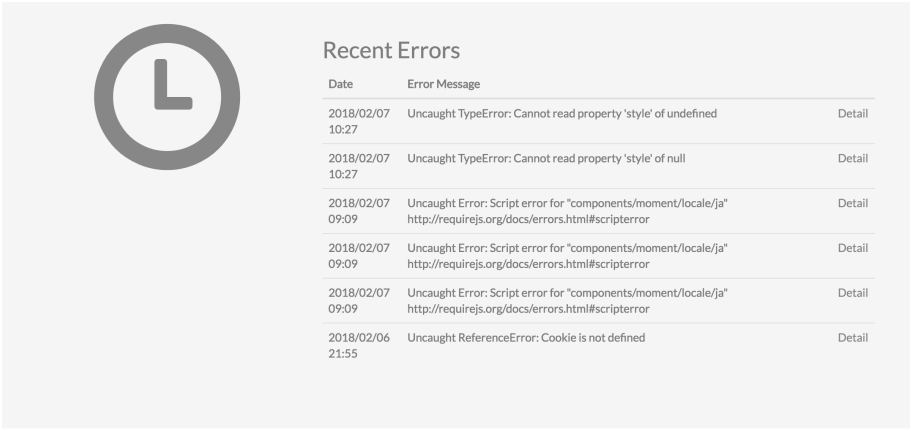


図 7: 公開ページ トップ画面



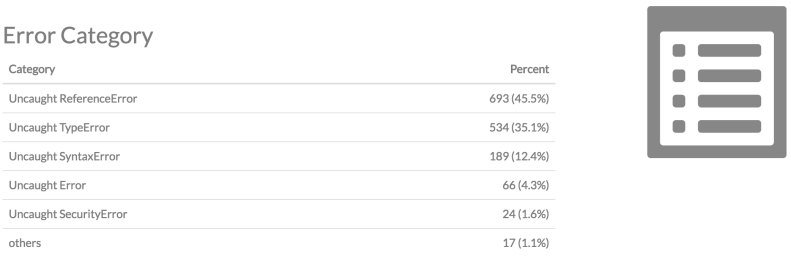


図 8: 公開ページ 最近のエラーとカテゴリの分布

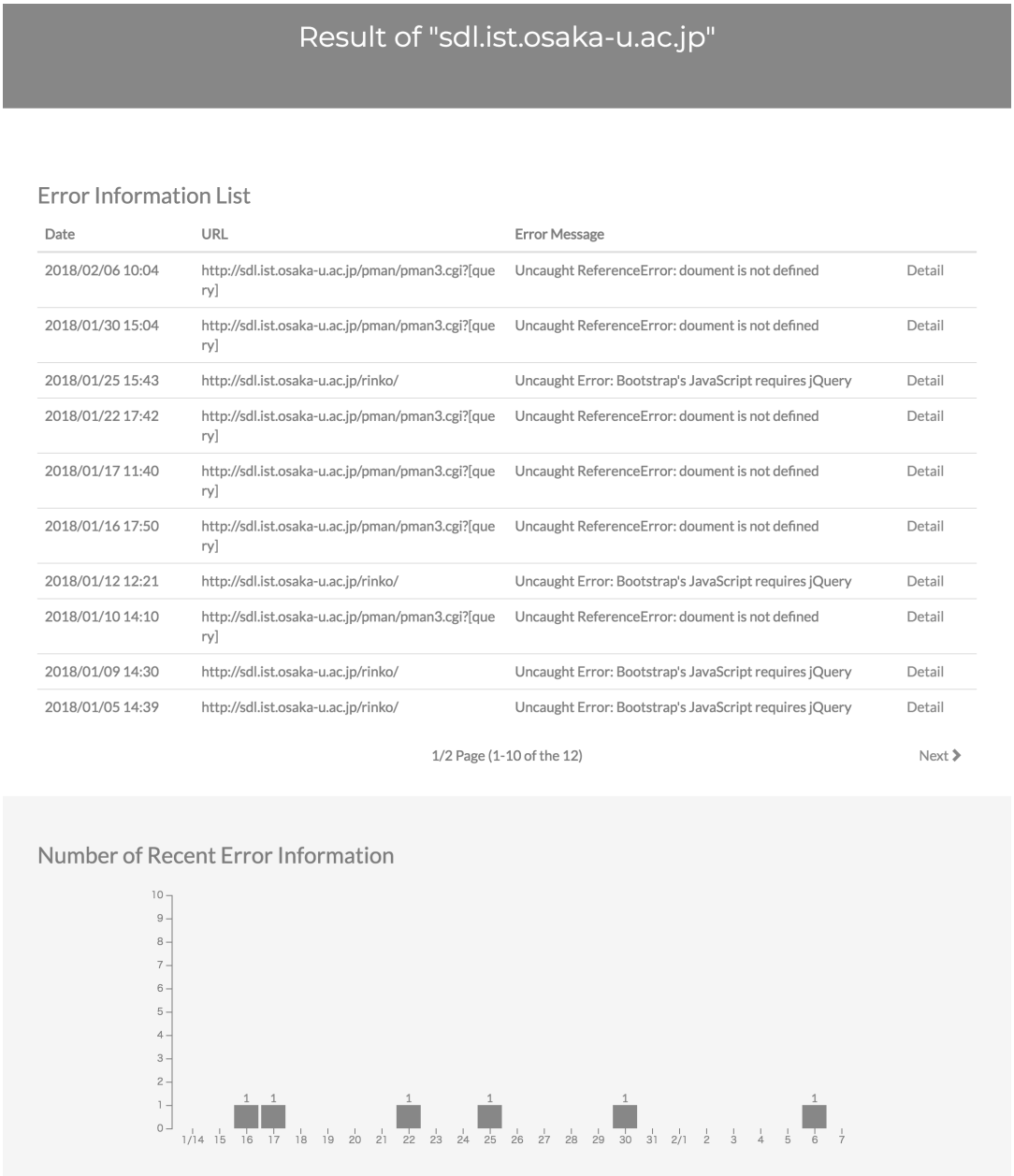


図 9: 公開ページ 検索結果画面

図 10: 公開ページ エラー情報詳細画面

被験者実験で実施アンケートとその結果

被験者実験において実施したアンケートについて，設問内容とその結果について表および図を記載する．それぞれの表・図は 7. 考察の章の説明時に適宜参照される．

表 4: アンケート内容

質問	回答方式
プラグインに関する質問	
プラグインのインストール後，Web ページの表示にかかる時間が長くなりましたか？	5 段階の選択式（全くそう思わない, そう思わない, どちらともいえない，そう思う，強くそう思う）
プラグインの送信済みエラー情報閲覧画面を何回確認しましたか？	5 段階の選択式（0 回，1〜3 回，4〜10 回，11〜30 回，それ以上）
エラー情報の中に，アカウント ID を含む URL のような，個人が限定できる項目が存在することがあります．存在することに抵抗を感じますか？	5 段階の選択式（全くそう思わない, そう思わない, どちらともいえない，そう思う，強くそう思う）
上の質問で「そう思う」または「強くそう思う」と回答した方にお訊きします．具体的にどのような点に抵抗を感じますか？	自由回答
エラー情報収集に協力するユーザにとって，プラグインにあったらいいと思う機能はありますか？	自由回答
公開ページに関する質問	
実験中に公開ページを何回閲覧しましたか？	5 段階の選択式（0 回，1〜3 回，4〜10 回，11〜30 回，それ以上）
現在公開ページでは，URL によって検索をして該当するエラー情報一覧を表示するという機能があります．ご自身がエラーリポジトリを利用する開発者であるとして，あったらいいと思う機能はありますか？	自由回答

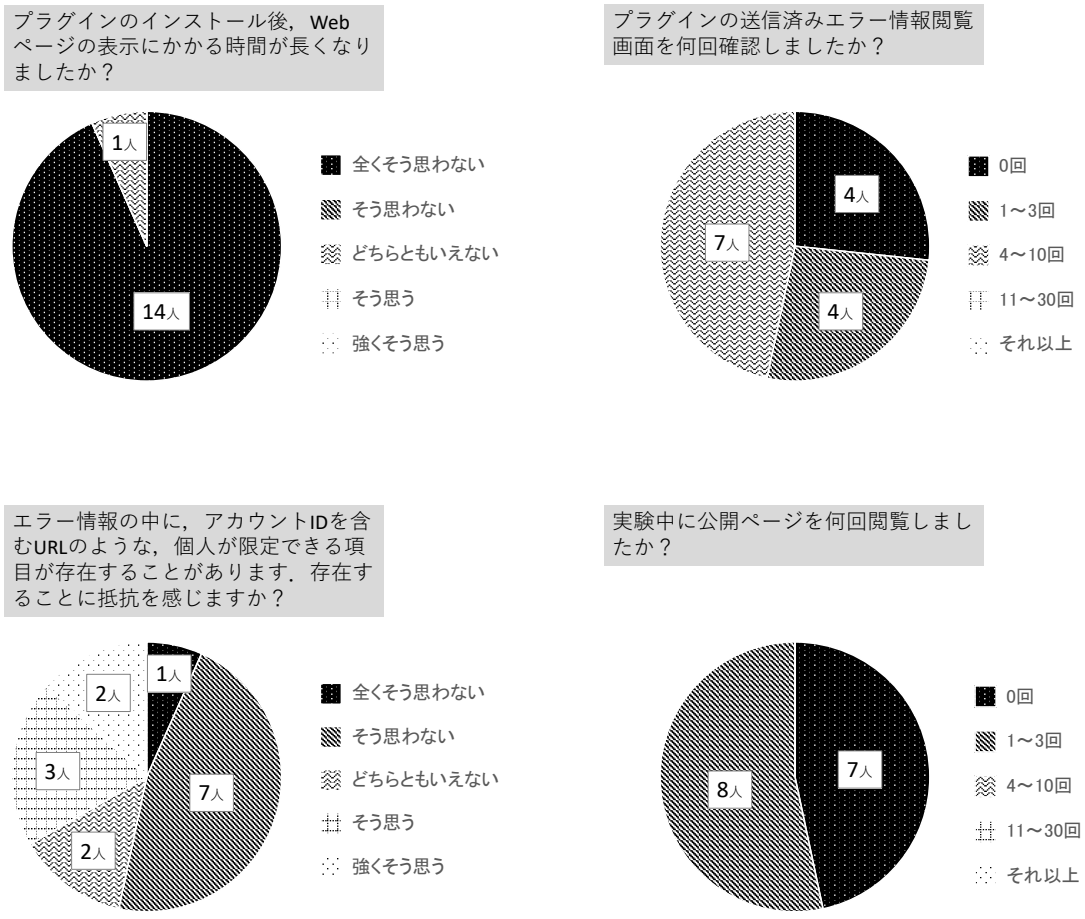


図 11: アンケート結果 選択式の割合

アカウント **ID** 等の公開について具体的に抵抗を感じる点についての回答まとめ

- 匿名化や統計処理されていない情報を公開している点
- SNS 等の趣味に使うアカウントを特定されたりする点
- URL を知っていれば誰でも閲覧できる請求書等の URL が漏れることで，住所や氏名が漏れてしまう点

プラグインにあったら良いと思う機能の回答まとめ

- エラー情報送信時に，バッヂやオーバーレイ表示などで収集した旨（＋収集した情報の概要）を通知する機能
- 金銭的なインセンティブ
- 公開ページのリンク

公開ページにあったら良いと思う機能の回答まとめ

- エラーメッセージによる検索機能
- エラーの改善案を提案する機能
- エラーの発生を開発者に自動的に通知する機能
- 各エラー情報についてデベロッパが議論できる機能

参考文献

[1] Jatinder Manhas. Comparative study of cross browser compatibility as design issue in various websites. *Our Major Indexing at International Level*, Vol. 4852, p. 815, 2015.

[2] Matthew Allen. Web 2.0: An argument against convergence. In *Media Convergence and Deconvergence*, pp. 177–196. Springer, 2017.

[3] Tõnis Saar, Marlon Dumas, Marti Kaljuve, and Nataliia Semenenko. Browserbite: Cross-browser testing via image processing. *Software: Practice and Experience*, Vol. 46, No. 11, pp. 1459–1477, 2016.

[4] Ali Mesbah and Mukul R. Prasad. Automated cross-browser compatibility testing. In *Proceedings of the International Conference on Software Engineering*, pp. 561–570, 2011.

[5] Ping Chen, Nick Nikiforakis, Christophe Huygens, and Lieven Desmet. A dangerous mix: Large-scale analysis of mixed-content websites. In *Information Security*, pp. 354–363. Springer, 2015.

[6] Euyshick Hong and Jun Kim. Drsax. js: A javascript based unified web audio library and framework. In *Proceedings of the International Conference on Algorithms, Computing and Systems*, pp. 74–78, 2017.

[7] Jeffrey A. Burke, Deborah Estrin, Mark Hansen, Andrew Parker, Nithya Ramanathan, Sasank Reddy, and Mani B. Srivastava. Participatory sensing. *Center for Embedded Network Sensing*, pp. 1–5, 2006.

[8] Delphine Christin, Andreas Reinhardt, Salil S. Kanhere, and Matthias Hollick. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software*, Vol. 84, No. 11, pp. 1928–1946, 2011.

[9] Frolin S. Ocariza Jr, Karthik Pattabiraman, and Benjamin Zorn. Javascript errors in the wild: An empirical study. In *Proceedings of the International Symposium on Software Reliability Engineering*, pp. 100–109, 2011.

[10] EllieD' Hondt, Matthias Stevens, An Jacobs. Participatory noise mapping works! an evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring. *Pervasive and Mobile Computing*, Vol. 9, No. 5, pp. 681–694, 2013.

[11] Min Mun, Sasank Reddy, Katie Shilton, Nathan Yau, Jeff Burke, Deborah Estrin, Mark Hansen, Eric Howard, Ruth West, and Péter Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pp. 55–68, 2009.

[12] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. Cartel: A distributed mobile sensor computing system. In *Proceedings of the International Conference on Embedded Networked Sensor Systems*, pp. 125–138, 2006.

[13] Prabal Dutta, Paul M. Aoki, Neil Kumar, Alan Mainwaring, Chris Myers, Wesley Willett, and Allison Woodruff. Common sense: Participatory urban sensing using a network of handheld air quality monitors. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, pp. 349–350, 2009.

[14] Pengfei Zhou, Yuanqing Zheng, and Mo Li. How long to wait?: Predicting bus arrival time with mobile phone based participatory sensing. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pp. 379–392, 2012.

[15] Stephen P Tarzia, Peter A Dinda, Robert P Dick, and Gokhan Memik. Indoor localization without infrastructure using the acoustic background spectrum. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pp. 155–168, 2011.

[16] Yohan Chon, Nicholas D. Lane, Fan Li, Hojung Cha, and Feng Zhao. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *Proceedings of the ACM Conference on Ubiquitous Computing*, pp. 481–490, 2012.

[17] 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.

[18] Juong-Sik Lee and Baik Hoh. Dynamic pricing incentive for participatory sensing. *Pervasive and Mobile Computing*, Vol. 6, No. 6, pp. 693–708, 2010.

[19] Irene Garcia Martí, Luis E. Rodríguez, Mauricia Benedito, Sergi Trilles, Arturo Beltrán, Laura Díaz, and Joaquín Huerta. Mobile application for noise pollution monitoring through gamification techniques. In *Proceedings of the International Conference on Entertainment Computing*, pp. 562–571, 2012.

[20] Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 1932.

[21] Michael S. Matell and Jacob Jacoby. Is there an optimal number of alternatives for likert scale items? study i: Reliability and validity. *Educational and Psychological Measurement*, Vol. 31, No. 3, pp. 657–674, 1971.

[22] Yoshitaka Ueyama, Morihiko Tamai, Yutaka Arakawa, and Keiichi Yasumoto. Gamification-based incentive mechanism for participatory sensing. In *Proceedings of the Pervasive Computing and Communications Workshops*, pp. 98–103, 2014.

[23] Chuan Yue and Haining Wang. Characterizing insecure javascript practices on the web. In *Proceedings of the International Conference on World Wide Web*, pp. 961–970, 2009.

[24] Ravi Chugh, Jeffrey A. Meister, Ranjit Jhala, and Sorin Lerner. Staged information flow for javascript. *ACM Sigplan Notices*, Vol. 44, No. 6, pp. 50–62, 2009.

[25] Google. Robots.txt Spesifications. Available online "`https://developers.google.com/search/reference/robots_txt`" (January 2018).

[26] Ueli M. Maurer. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory*, Vol. 39, No. 3, pp. 733–742, 1993.

[27] Rakesh Pai. Errorception. Available online "`https://errorception.com`" (October 2017).

[28] Airbrake. Airbrake. Available online "`https://airbrake.io`" (October 2017).