자바스크립트 기반 실제 웹 어플리케이션의 오류 조사

안재민, 김문주

한국과학기술원 전산학과
대전광역시 유성구 구성동 373-1
{jaemin, moonzoo}@cs.kaist.ac.kr

요약: 동적 웹 페이지를 구현하기 위해 클라이언트 사이드 자바스크립트 기반 웹 어플리케이션이 많이 쓰인다. 이로 인해 자바스크립트 어플리케이션의 정확성, 보안성, 성능 등을 보장하기 위한 테스트가 중요해졌다. 하지만 자바스크립트의 고유한 특성으로 인해 기존의 테스트 기법은 자바스크립트 어플리케이션 테스트에 적합하지 않고, 기존의 테스트 기법을 통해 테스트 과정을 마친 자바스크립트 어플리케이션은 여전히 많은 버그를 내포한 채 서비스되기도 한다. 이 논문은 실제 자바스크립트 기반 웹 어플리케이션의 오류 추적 시스템 (Bug Tracking System)을 통해 자바스크립트 어플리케이션의 오류에 대해 조사한다. 4 개의 공개 웹 어플리케이션(WordPress, BugZilla, OpenOffice, and MapTools)을 대상으로 오류 조사를 시행하였고, 한 어플리케이션 당 25 개의 오류를 조사했다. 조사한 오류를 그 영향, 원인, 수정 방법을 기준으로 분류했다.

핵심어: 자바스크립트, 웹 어플리케이션, 버그 분류

1. 서 론

최근의 웹 어플리케이션은 웹 2.0 표준을 만족시키고, 웹 페이지를 동적으로 표현하기 위해 자바스크립트를 사용해 페이지 내용을 변화시킨다. 이러한 웹 어플리케이션은 보다 나은 접근성을 가진다. 자바스크립트는 클라이언트에서 실행되며, 웹 페이지를 표현하는데 쓰이는 Document Object Model (DOM)을 수정해 웹 페이지의 동적 변화를 만든다[1]. 따라서, 자바스크립트의 정확성, 안정성, 보안성 등은 전체 웹 어플리케이션의 정확성, 안정성, 보안성 등과 직결된다.

자바스크립트[2]는 1995 년에 고안되었지만, 2000 년대 중반부터 본격적으로 사용되었고, 따라서 자바스크립트 어플리케이션의 효과적이고, 효율적인 테스트 기법을 소개하는 연구는 2000 년대 중반 이후 시작되어 그 수는 그리 많지 않다[3-6]. 그간의 자바스크립트 테스트 기법의 연구 성과에도 불구하고, 많은 수의 실제 서비스 중인 자바스크립트 기반 웹 어플리케이션이 오류를 가지고 있다[7]. 이러한 오류를 방지하고 보다 나은 테스트 기법의 개발을 위해서는 자바스크립트 오류의 특성에 대해 조사할 필요가 있다. 자바스크립트와 그 오류의 특성에 기반해 개발 방법론이나 테스트 기법 등을 개발하면 그 효과성이 더욱 뛰어날 것이다.

자바스크립트 기반 웹 어플리케이션 테스트 기법에 관한 연구[3-6]에서는 기존의 C, C++, Java 어플리케이션을 대상으로 하는 테스트 기법을 자바스크립트 기반 웹 어플리케이션에 적용하고 있다.

자바스크립트 기반 웹 어플리케이션의 동적 및 정적 특성에 대해 연구한 사례도 있다. 하지만 이러한 연구는 자바스크립트 어플리케이션의 테스트나 오류 수정 등에 초점을 맞추지는 않았다. 자바스크립트의 특성에 대한 연구는 대부분 보다 나은 자바스크립트 엔진을 개발과 자바스크립트의 실행 성능에 중점을 두거나[8,9] 자바스크립트 어플리케이션의 보안 문제와 관련된 특성을 다루고 있다[10,11].

본 연구의 주 목적은 자바스크립트 기반 웹 어플리케이션 오류의 특성에 대해 조사하는 것이다. 오류가 어플리케이션의 실행에 미치는 영향, 오류의 발생 원인, 그리고 발견된 오류를 수정하는 방법 등을 조사한다. 이를 통해 자바스크립트 기반 웹 어플리케이션을 개발하는 과정에서 오류를 줄일 수 있는 방법과 효과적으로 자바스크립트 기반 웹 어플리케이션을 테스트하는 방법을 만들 수 있을 것이다.

이 논문에서 조사한 자바스크립트 오류는 공개된 오류 추적 시스템 (Bug Tracking System)을 기반으로 하고 있다.

본 논문에서는 WordPress, BugZilla, OpenOffice, MapTools 의 4 개의 공개 어플리케이션을 대상으로 조사했다. 위 어플리케이션은 충분한 수의 오류 보고를 포함하는 오류 추적 시스템을 공개하고 있는 자바스크립트 기반 웹 어플리케이션이다. 우리는 보고된 오류 중 개발자가 실제 오류로 확인하고, 수정한 오류만을 대상으로 했다. 하나의 어플리케이션 당 100~300 개 가량의 위 조건에 맞는

오류 중 25 개의 오류를 임의로 선택해 총 100 개의 오류 조사했다. 오류를 수집한 후, 우리는 오류가 (1) 어플리케이션에 미치는 영향, (2) 발생 원인, (3) 수정 방법 등을 조사하고 이를 기준으로 분류했다.

본 연구의 주요 기여는 다음과 같다.

(1) 실제 자바스크립트 기반 웹 어플리케이션 오류 조사

(2) 3 가지 기준을 사용한 실제 자바스크립트 기반 웹 어플리케이션 오류의 분류

(3) 자바스크립트 기반 웹 어플리케이션 개발자, 테스터 등을 위한 활용 방안 제시

본 논문의 구성은 다음과 같다. 2 장에서는 자바스크립트와 자바스크립트 오류에 대한 배경지식을 소개한다. 3 장에서는 자바스크립트 테스트 기법, 특성 조사에 관한 관련 연구 및 오류 추적 시스템을 통한 조사 방법에 관한 관련 연구를 소개한다. 4 장에서는 조사 대상 자바스크립트 어플리케이션과 조사 방법에 대해 설명한다. 5 장부터 8 장에서는 조사 및 분류 결과를 소개한다. 9 장에서는 조사 및 분석 결과의 활용방안을 설명한다. 마지막으로 10 장에서 결론을 맺고 추후 연구 방향에 대해 소개한다.

2. 배경 지식

자바스크립트는 현재 웹 어플리케이션에서 중추적인 역할을 수행한다[12]. [13]에 따르면 가장 많이 방문되는 100 개의 웹 사이트 중 97 개의 웹 사이트에서 자바스크립트를 사용하고 있다. 자바스크립트는 기존의 C, C++, Java 등과 구분되는 고유의 특성이 있다. 예를 들어 자바스크립트는 동적 타입을 사용하고, 변수 테이블을 동적으로 관리하며, 실행 도중 새로운 자바스크립트 코드를 만들어 실행할 수도 있다. 이러한 고유 특성으로 인해 자바스크립트 고유의 오류가 발생한다.

일반적인 웹 어플리케이션은 HTML, CSS, 스크립트 언어(i.e. 자바스크립트) 의 3 가지 요소로 구성된다. HTML 은 웹 페이지의 전체적인 모양을 결정하고, CSS 는 웹 페이지의 레이아웃 등을 정의한다. 스크립트 언어는 웹 페이지의 주요 기능을 구현하기 위해 사용된다. 스크립트 언어는 HTML 에 직접 삽입되거나 다른 스크립트 언어의 실행을 통해 DOM 에 동적으로 삽입될 수 있다. 앞서 언급했듯이 자바스크립트는 현대의 웹 어플리케이션에서 가장 많이 사용되는 스크립트 언어기 때문에 자바스크립트의 오류는 전체 웹 어플리케이션의 오류로 직결된다.

대부분의 웹 어플리케이션은 이벤트를 기반으로 구현되어 있다. 따라서 웹 어플리케이션은 많은 수의 이벤트 처리자(event handler)를 가지고 있고 이는 자바스크립트로 구현된다. 이러한 이벤트 기반 웹 어플리케이션에서 사용되는 자바스크립트는 연속적으로 실행되지 않는다. 이는 자바스크립트 테스트를 더욱 어렵게 한다.

자바스크립트는 HTML 에 직접 삽입되거나 동적으로 다른 자바스크립트의 실행을 통해 삽입될 수 있다. 삽입 방법과 무관하게 DOM 에 자바스크립트가 추가되면 자바스크립트 엔진은 추가된 자바스크립트 코드를 실행하게 된다. 이 실행은 함수 및 변수의 정의, 이벤트 처리자의 등록 등이 될 수 있다. 자바스크립트의 최초 실행과정에서 실행에 앞서 추가된 자바스크립트 코드가 문법에 맞게 작성되었는지 확인하게 되는데 이 과정에서 문법 오류가 검출될 수 있다. 하지만 자바스크립트의 유연성으로 인해 몇몇의 문법 오류는 무시되고 실행될 수 있고 이는 예측할 수 없는 오류를 야기할 수 있다.

HTML 문서가 모두 처리된 이후에는 웹 페이지의 onload 이벤트가 발생한다. 만약 onload 이벤트의 처리자가 HTML 처리 과정에서 등록되어 있다면 해당 자바스크립트 처리자가 실행된다. onload 이벤트와 마찬가지로 다른 이벤트도 발생하는 시점에 이벤트 처리자가 등록되어 있다면 해당 처리자가 실행된다. 이벤트 처리자의 등록은 DOM 을 통해 이루어지는데 자바스크립트 실행은 DOM 을 수정할 수 있기 때문에 이벤트 처리자를 등록 또는 해제할 수 있다.

만약 특정 이벤트에 적절한 이벤트 처리자가 등록되어 있지 않으면 사용자 입력이 무시될 수 있다. 이는 응답 없음 오류를 발생시킬 수 있다. 또한 잘못된 이벤트 처리자가 등록되어 있을 경우에는 개발자 및 사용자의 기대와 다른 결과를 주는 잘못된 결과 오류를 발생시킬 수 있다.

자바스크립트는 또한 외부 이미지 등을 불러오거나 외부 어플리케이션을 실행시킬 수 있다. 하지만 이 때 외부 어플리케이션이나 자료 등을 사용할 수 없다면 전체 어플리케이션의 동작이 중단되어 비정상 종료 오류를 발생시키거나 외부 자료를 불러오는데 오랜 시간이 걸려 응답 없음 오류나 성능 오류를 발생시킬 수 있다.

3. 관련 연구

자바스크립트의 테스트 기법[3-6]에 대한 연구는 꾸준히 있어왔지만, 상대적으로 적은 연구자만이 자바스크립트 오류의 다양한 특성에 대해 연구했다.

Saxana 등은 자바스크립트를 대상으로 자동화된 테스트 기법을 제시하고 Kudzu 라는 도구를 개발했다[6]. Kudzu 는 임의 테스트 입력 생성 기법을 통해 자바스크립트의 다양한 이벤트 조합을 살펴볼 수 있게 했고, 심볼릭 실행을 통해 다양한 입력값을 살펴볼 수 있게 했다. Saxana 등이 사용한

기법은 일반적인 어플리케이션을 대상으로 한 테스트 기법들이다. 하지만 웹 어플리케이션은 보통 문자열을 입력값으로 갖고, 문자열을 사용하는 심볼릭 실행은 오래 걸리는 문제점이 있다. 때문에 Kudzu 는 테스트를 수행하는데 오랜 시간이 필요하다는 단점을 가지고 있다.

Artzi 등 또한 자동화된 테스트 프레임워크를 소개했다[3]. Artzi 등의 프레임워크는 피드백을 기반으로 테스트 입력을 생성하고 우선순위를 정한다. Artzi 등은 프레임워크를 통해 자바스크립트 어플리케이션의 70% 구문 커버리지를 달성하는 것을 보였다. 하지만 일반적인 어플리케이션의 구문 커버리지 요건을 고려하면 70%라는 수치는 상대적으로 그 효과가 낮다고 할 수 있다.

위에 소개한 테스트 기법의 한계는 자바스크립트 기반 웹 어플리케이션에 대한 부족한 이해일 수 있다. 이는 많은 연구자로 하여금 자바스크립트의 특성에 대해 연구하게 했다.

Richard 등은 자바스크립트 기반 어플리케이션의 여러 가지 특성을 측정했다[9]. Richard 등은 특히 eval 함수의 호출 회수 등 자바스크립트 기반 웹 어플리케이션의 동적 특성을 측정했다. 또한 Richard 등은 자바스크립트 벤치마크(i.e. SunSpider, V8)에 대해서도 그 특성을 측정하고 비교한다. 하지만 이 연구는 자바스크립트의 오류와 연관해 자바스크립트의 특성을 설명하지는 않는다.

Rataanaworabhan 등은 자바스크립트 벤치마크와 실제 웹 어플리케이션의 특성을 비교했다[8]. 비교결과는 고유 함수의 수, 함수 호출회수 등 동적, 정적 특성 모두 포함하고 있다. 또한 이벤트와 이벤트 처리자, 죽은 코드 등도 비교 대상이다. 하지만 이 연구 역시 자바스크립트 오류와 연관해 그 특성을 설명하고 있지는 않다.

위의 두 연구는 자바스크립트의 실행 특성을 분석하고 있지만 오류와 연관된 설명이 없어 신뢰성 있는 자바스크립트 개발 방법이나 테스트 기법 개발 등에 사용할 수 있는 결과는 아니다.

Ocariza 등은 웹 어플리케이션의 실행 도중의 콘솔 메시지를 기반으로 오류를 조사했다[7]. 결과에 따르면 실제 서비스중인 웹 어플리케이션 하나당 평균 4 개의 오류가 내재되어 있다. 하지만 콘솔 메시지를 기반으로 하기 때문에 실제 오류가 미치는 영향이나 원인 등이 누락되어 있다.

본 연구는 Jin 등의 연구[14] 방법을 따라 자바스크립트 오류의 특성을 오류 추적 시스템을 통해 조사한다. Jin 등은 성능 오류를 대상으로 오류 추적 시스템에서 오류를 수집하고 분석하여 성능 오류를 그 특징별로 분류했다. 또한 나아가 Jin 등은 조사 결과로부터 성능 오류를 찾기 위한 효과적인 테스트 기법도 소개하고 있다.

4. 조사 대상 및 방법

본 연구에서는 WordPress[1], BugZilla[2], OpenOffice[3], MapTools[4]등 4 가지 종류의 자바스크립트 기반 웹 어플리케이션을 대상으로 한다.

WordPress 는 웹 블로그 제작을 도와주는 공개 웹 어플리케이션이다. WordPress 의 주요 기능은 자바스크립트를 통해 구현되어 있고 2003 년부터 서비스를 시작해 충분한 수의 오류가 보고되어 있다.

Bugzilla 는 오류 추적 시스템을 구축하기 위한 공개 웹 어플리케이션이다. Mozilla 재단의 어플리케이션의 오류 추적 시스템 구축을 위해 구현되어 공개되었다. 많은 공개 소프트웨어의 오류 추적시스템은 Bugzilla 를 통해 구현되어 있고, 충분한 수의 오류가 보고되어 있다.

OpenOffice 는 웹 환경에서 워드프로세서, 스프레드시트 등의 문서작업을 하기 위한 페이지 구축을 위한 공개 웹 어플리케이션이다.

MapTools 는 웹 환경에서 지도와 관련한 웹 페이지를 만들기 위한 공개 웹 어플리케이션이다. MapTools 는 지도 서버 구축, 웹 페이지 클라이언트 구현 등을 위한 다양한 요소로 나뉘어져 있다. 본 연구에서는 웹 페이지에 지도를 보여주고 다양한 사용자 인터페이스를 제공하는 Chameleon 요소를 대상으로 한다.

WordPress 의 경우에는 독자적인 방법으로 오류 추적 시스템을 운영하고 있다. 반면에 WordPress 를 제외한 다른 어플리케이션의 경우에는 Bugzilla 를 기반으로 오류 추적 시스템을 구축해 운영하고 있다. 그림 1 과 그림 2 는 각각 WordPress 의 오류 추적 시스템과 Bugzilla 기반의 오류 추적 시스템이다.

본 연구에서는 개발자가 실제 오류라고 확인하고 수정한 오류만을 대상으로 조사한다. 개발자가 실제 수정한 오류는 어플리케이션의 기능에 있어 치명적이라고 판단했기 때문이다. 또한 개발자가 수정하지 않은 오류에 대해서는 추후 분석 과정에서 개발자가 어떻게 오류를 수정했는지 알 수 없기 때문이다. 2005 년 이전에 보고된 오류 또한 배제했다. 보고된 후 시일이 많이 지난 오류는 현대의 자바스크립트 오류의 특성과 다소 차이가 있을 수 있기 때문이다.

각각의 어플리케이션에서 개발자가 수정한 2006 년 이후의 오류 중 임의로 25 개씩의 오류를 추출해 총 100 개의 오류 보고를 대상으로 조사했다.

---

[1] http://core.trac.wordpress.org/report

[2] https://bugzilla.mozilla.org/

[3] https://issues.apache.org/ooo/

[4] http://bugzilla.maptools.org/

ID	Product	Comp	Assignee	Status	Resolution ▲	Summary	Changed
2768	word pro	ui	requirements	CONF	---	load html folders as root for website under OO	2010-05-21
4638	word pro	code	Mathias_Bauer	CONF	---	ligature support	2011-01-30
5014	installa	ui	requirements	ACCE	---	no java/script support installed: doesn't list components requiring java	2010-05-21
10274	database	code	frank.schoenheit	CONF	---	LDAP datasource cannot handle all attributes in inetOrgPerson	2008-03-28
16036	presenta	code	augustus.saunders	ACCE	---	Animations in Macromedia Flash Export	2011-05-24
21148	framewor	code	erwin.tenhumberg	CONF	---	Q-PCD Programmability-3: Modular IDE	2004-05-26
23786	ucb	code	tobias.krause	ACCE	---	Unable to open documents hosted by a certain web application	2007-08-20
25264	framewor	scriptin	ab	ACCE	---	Console needed for script related logging	2006-05-17
25275	framewor	scriptin	ab	ACCE	---	It should be possible to disable and uninstall the default Script Providers	2006-05-17
30616	framewor	scriptin	ab	ACCE	---	Libraryname "Highlight" not associative to its content	2006-05-09
34625	spreadsh	save-exp	requirements	CONF	---	Export spreadsheet as javascript	2010-09-12
41434	gsl	code	gsldev	CONF	---	PDF export : add dynamic fields like date	2011-08-10
44628	ui	ui	Mathias_Bauer	CONF	---	Hello World library descriptions are not clear	2005-03-16
50526	framewor	scriptin	ab	ACCE	---	ScriptOrganizer: Cannot rename Library	2006-05-17
50527	framewor	scriptin	ab	ACCE	---	ScriptOrganizer: Delete-Button is enabled for document	2006-05-17
50529	framewor	scriptin	ab	ACCE	---	ScriptOrganizer: Scripts always get suffix	2006-05-18
54752	framewor	ui	requirements	UNCO	---	Add Commands to call JavaScript, BeanShell, Python organizers	2008-05-16
54753	framewor	ui	ab	ACCE	---	JavaScript share libraries are editable	2006-05-19
56709	framewor	scriptin	ab	ACCE	---	Using SDK Developersguide result into not saving embedded Scripting Framework Macros	2006-06-19

[그림 1] Bugzilla, OpenOffice, MapTools 등 Bugzilla 를 기반으로 한 오류 추적 시스템

Ticket	Summary	Owner	Component	Priority	Severity	Milestone	Type	Workflow	Modified
#21334	Accessibility of Quick Edit panel in Posts/Pages/etc		Accessibility	normal	normal	Awaiting Review	defect (bug)		07/24/12
#16103	Blue Admin theme doesn't have enough contrast		Accessibility	normal	normal	Future Release	defect (bug)	has-patch	12/07/12
#22682	Close Button Breaks in Customizer After Refresh		Accessibility	low	minor	Future Release	defect (bug)		12/04/12
#22606	Customizer: "Select a file" link cannot be opened from keyboard		Accessibility	normal	normal	Awaiting Review	defect (bug)		12/04/12
#15926	Give header and background images alt tags		Accessibility	normal	minor	Future Release	defect (bug)	has-patch	03/05/12
#20880	Keyboard navigation in Appearance > Header is broken		Accessibility	normal	normal	Awaiting Review	defect (bug)		06/08/12
#12825	Largest minimum text size in FF prefs makes admin display terrible		Accessibility	normal	normal	Future Release	defect (bug)		07/18/12
#20294	User profile edit: no label on second password field		Accessibility	normal	normal	Awaiting Review	defect (bug)	has-patch	10/04/12
#18801	Accessibility Enhancements to Settings API	taupecat*	Accessibility	normal	normal	Awaiting Review	enhancement	has-patch	08/07/12
#18900	Add a few more hide-if-no-js classes		Accessibility	normal	minor	Awaiting Review	enhancement	has-patch	09/28/12
#15080	Comment Form Should use HTML5 input types for better accessibility		Accessibility	normal	normal	Future Release	enhancement	has-patch	11/07/12
#16433	Extend function to optionally include commenter name in comment_reply_link	merty*	Accessibility	low	normal	Future Release	enhancement	has-patch	09/12/11
#14045	Give the menus page an accessibility mode option, like the widgets screen.		Accessibility	normal	normal	Future Release	enhancement	early	11/17/12
#18650	Make archives and categories widgets dropdown ada compliant		Accessibility	normal	normal	Awaiting Review	enhancement	dev-feedback	08/03/12
#15081	Search Form should use type='search'		Accessibility	lowest	minor	Future Release	enhancement	has-patch	12/05/10
#21414	Use the "Keyboard Shortcuts" checkbox in the user profile to turn on/off all custom shortcuts		Accessibility	normal	normal	Awaiting Review	enhancement		10/14/12

[그림 2] WordPress 의 오류 추적 시스템

5. 오류의 영향에 따른 분류

조사한 오류를 어플리케이션에 미치는 영향에 따라 분류했다.

- 잘못된 결과: 어플리케이션의 실행 결과가 개발자와 사용자의 기대와 다른 결과인 경우
- 비정상 종료: 어플리케이션의 실행이 멈춘 경우
- 응답 없음: 어플리케이션의 실행이 멈추지는 않았으나, 아무런 실행 결과를 보여주지 않는 경우
- 성능: 어플리케이션의 실행 결과가 오랜 시간이 지난 후 보여지는 경우
- 기타: 화면에 보여지는 요소의 충돌 등 위 4 가지 경우에 포함되지 않는 경우

표 1 과 그림 3 은 조사한 오류를 그 영향에 따라 분류한 것이다.

표 1 에서 볼 수 있듯 잘못된 결과를 보여주는 오류가 가장 많았다. 이 종류의 오류는 일반적인 기능 오류다.

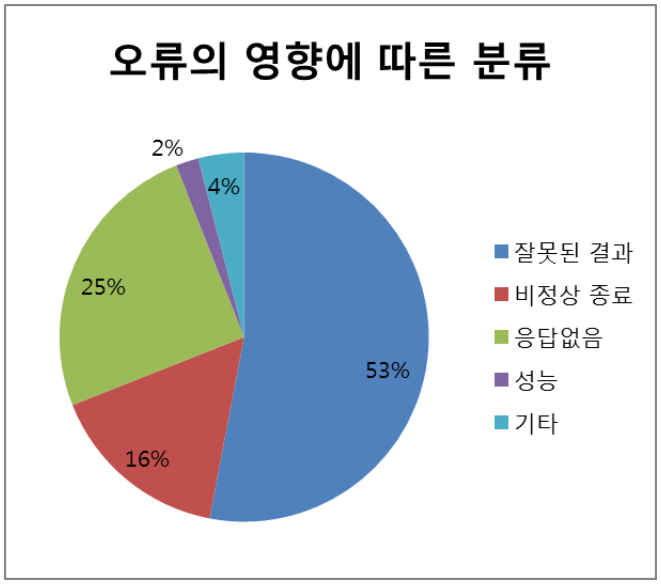
두 번째로 많은 오류의 영향은 응답 없음이었다. 이러한 오류는 대체로 자바스크립트 엔진의 유연성에서 기인했다. 예를 들어 오류 발생으로 인해 어플리케이션이 비정상 종료를 일으키고 실행이 멈추어야 하지만, 오류를 무시하고 계속 실행되는 것이다. 때문에 특정 이벤트 처리자의 실행이 무시되어 아무런 응답이 없게 되는 것이다. 또한 적절한 이벤트 처리자가 등록되지 않는 경우에도 이러한 오류가 발생했다.

[표 1] 오류의 영향에 따른 분류

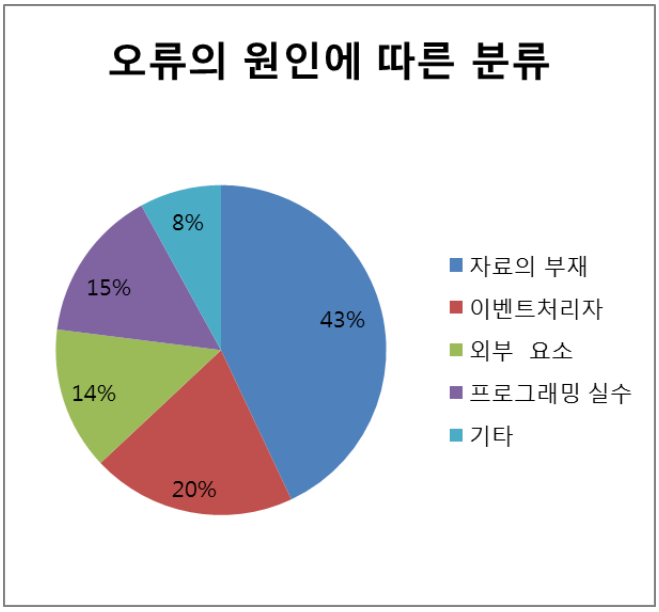
오류의 영향	Word Press	BugZilla	Open Office	Map Tools	총
잘못된 결과	10	7	15	21	53
비정상 종료	5	3	5	3	16
응답 없음	8	12	5	0	25
성능	1	0	0	1	2
기타	1	3	0	0	4

[표 2] 오류의 원인에 따른 분류

오류의 원인	Word Press	BugZilla	Open Office	Map Tools	총
자료의 부재	13	6	7	17	43
이벤트 처리자	6	10	1	3	20
외부 요소	1	3	8	2	14
프로그래밍 실수	5	3	5	2	15
기타	0	3	4	1	8



[그림 3] 오류의 영향에 따른 분류



[그림 4] 오류의 발생 원인에 따른 분류

비정상 종료를 일으키는 오류는 상대적으로 쉽게 발견된다. 하지만 자바스크립트 실행의 경우의 수가 다양해 테스트 과정에서 모두 수정하지 못 한 것으로 보인다.

성능 저하를 일으키는 오류는 불필요한 연산이 많이 포함되거나 필요한 자료에 접근할 때 발생하는 대기시간으로 인해 주로 발생한다. 예를 들어 자동 완성 기능을 구현할 때, 매 입력마다 자동 완성과 관련된 함수가 호출되어 입력 사이에 지연 시간이 발생해 불편을 초래하는 오류가 있었다.

기타 오류로는 광고와 관련된 오류, 웹 페이지의 레이아웃이 겹쳐, 일부 내용을 가리는 오류 등이 있었다. 광고와 관련된 오류는 광고 부분의 자바스크립트 코드에서 호출되는 함수가 same origin policy 를 어겨 발생했다. 웹 페이지의 레이아웃이 겹치는 경우 이미지의 위치 설정이 잘못되어 이미지가 웹 페이지의 일부 텍스트를 가리게 되어 발생했다.

6. 오류의 원인에 따른 분류

이 장에서는 조사한 오류의 발생 원인에 따라 분류한 결과를 소개한다.

- 자료의 부재: 이미지 등 자바스크립트 외부의 파일 등을 불러오는데 실패하는 경우
- 이벤트 처리자: 잘못된 처리자를 호출하거나 적절한 처리자가 등록되지 않은 경우
-기타: 사소한 프로그래밍 실수 등 위의 3 가지 경우에 포함되지 않는 경우

표 2 와 그림 4 는 오류의 발생 원인에 따라 분류한 것이다.

자료의 부재로 인한 오류는 파일 등의 자료에 접근하는데 실패하면서 발생한다. 자바스크립트에서 주로 접근하는 자료는 클라이언트에 저장되어 있는 경우도 있는데, 클라이언트의 상황이 확실하지 않기

때문에 이 오류가 빈번하게 발생한다. 많은 경우 자바스크립트 상에서 필요한 파일 등의 존재 유무를 확인하지 않은 채 접근하고, 이는 오류로 직결된다. 또한 해당 파일이 있더라도 파일 이름만 같고 내용이 다를 경우에도 오류가 발생한다. 이 원인으로 발생한 오류는 응답 없음 혹은 잘못된 결과를 초래한다.

이벤트 처리자로 인한 오류는 잘못된 처리자를 호출하거나 처리자를 호출하지 않는 경우다. 이 원인으로 인한 오류는 주로 어플리케이션의 업데이트 직후에 보고된다. 개발자가 자바스크립트 코드를 수정하면서 이벤트 처리자 함수 이름 등을 바꾸고 이벤트에 등록하는 부분을 수정하지 않아 발생한다. 결과적으로 이벤트가 발생해도 이벤트 처리자가 실행되지 않는 경우에는 응답 없음을 초래하고, 잘못된 처리자가 실행되는 경우에는 5 장에서 언급한 모든 종류의 오류 결과를 초래할 수 있다.

그림 5 는 잘못된 이벤트 처리자 등록의 예제다. 이 경우 window.onload 이벤트의 처리자는 disableRequest 함수로 덮어씌워진다. 하지만 개발자의 본래 의도는 disableRequest 함수를 window.onload 이벤트의 처리자 중 하나로 추가하는 것이다. 그림 6 은 외부 라이브러리를 사용해 올바르게 고친 것이다.

```
69: ……
70: window.onload=disableRequest;
71: ……
```

[그림 5] 잘못된 이벤트 처리자의 예

```
69: ……
70: YAHOO.util.Event.addListener
        (window, “load”, disableRequest);
71: ……
```

[그림 6] [그림 5]의 예제의 올바른 수정

또 다른 이벤트 처리자로 인한 오류의 예는 너무 많은 이벤트 처리자가 등록된 경우다. 예를 들어 자동 완성 기능을 사용할 때 매 키보드 입력마다 이벤트 처리자가 불려 자동완성과 관련된 부분을 실행할 경우 성능이 저하되는 문제가 발생한다.

외부요소로 인한 오류는 Java Servlet 등 외부 어플리케이션이나 라이브러리를 사용할 때 문제가 있을 경우 발생한다. 대부분의 외부 요소로 인한 오류는 전체 어플리케이션의 실행을 멈추게 하는 비정상 종료 상황을 발생시킨다.

그림 7 은 외부 데이터베이스를 사용할 때 발생하는 오류의 예제다. 58 번째 줄의 group_id 가 실제로는 group.id 가 되어야 한다. 올바르지 않은 SQL 요청을 데이터베이스에 보내고 데이터베이스 서버는 이 요청을 처리하지 못하고 오류가 일어나게 된다. 외부 데이터베이스의 오류의 영향으로 자바스크립트 어플리케이션도 비정상적으로 종료되게 된다.

```
55: …..
56: SendSQL(“SELECT DISTINCT groups.id, isactive,”.
57: “oldmap.membercontrol, newmap.membercontrol”.
58: “CASE WHEN groups_id IN ($grouplist) THEN 1 ELSE 0
      END, ”.
59: “bug_group_map.group_id IS NOT NULL ”.
60: “FROM groups”.
61: …..
```

[그림 7] 외부 데이터베이스 사용 오류 예제

```
55: …..
56: SendSQL(“SELECT DISTINCT groups.id, isactive,”.
57: “oldmap.membercontrol, newmap.membercontrol”.
58: “CASE WHEN groups.id IN ($grouplist) THEN 1 ELSE 0
      END, ”.
59: “bug_group_map.group_id IS NOT NULL ”.
60: “FROM groups”.
61: …..
```

[그림 8] [그림 7]의 예제의 올바른 수정

변수 이름을 잘못 사용한 경우, 코드 상에 오타가 발생하는 경우 등 사소한 프로그래밍 실수로 인한 오류가 15 개가 존재했다.

그 외의 기타 원인으로는 이미지 등의 자료의 잘못된 사용, 함수 및 변수의 잘못된 덮어쓰기 등이 있다.

7. 오류의 수정 방법에 따른 분류

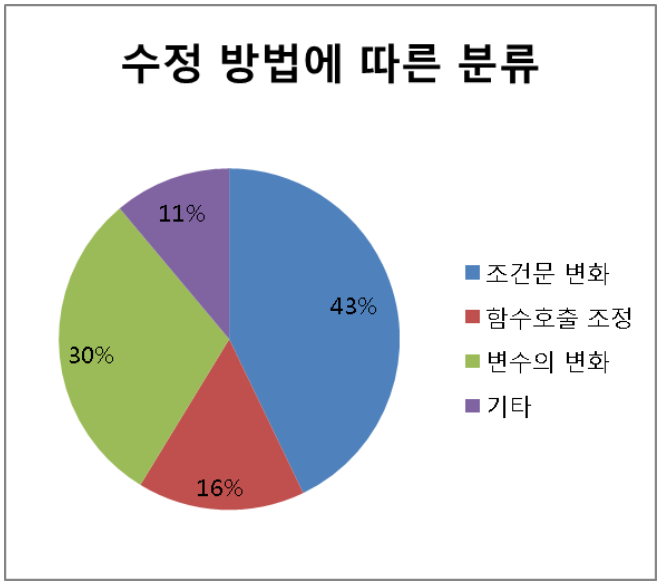
이 장에서는 개발자가 어떻게 오류를 수정했는지에 따라 조사한 오류를 분류한 결과를 소개한다. 하나의 오류를 수정하기 위해서 다양한 방법을 사용할 수 있기 때문에 하나의 오류가 두 개 이상의 분류에 포함될 수 있다.

- 조건문 변화: 조건문의 추가, 제거, 변경을 통해 오류를 수정하는 경우
- 함수 호출 조정: 함수 호출을 추가, 제거 혹은 순서의 변경 등을 통해 오류를 수정하는 경우
- 변수의 변화: 변수의 값, 함수 호출시 사용되는 parameter 의 값 등을 변경해 오류를 수정하는 경우
- 기타: 오타 수정 등 위 3 가지 방법 외에 다른 방법으로 오류를 수정하는 경우

표 3 과 그림 9 는 개발자가 오류를 수정하는 방법에 따른 분류다.

[표 3] 오류 수정 방법에 따른 분류

오류 수정방법	Word Press	BugZilla	Open Office	Map Tools	총
조건문 변화	14	9	10	21	54
함수호출 조정	6	5	7	2	20
변수의 변화	10	14	10	4	38
기타	4	2	3	5	14



[그림 9] 오류 수정 방법에 따른 분류

오류의 수정과정 중 54 개의 오류는 조건문이 변경되었다. 이는 조건문이 새로 추가, 제거, 혹은 변경되었다는 의미다. 조건문 변화는 주로 자료의 부재로 인한 오류를 수정하기 위해 자료가 사용가능한지 확인하는 조건문을 추가해 오류를 수정했다. 또한 이벤트 처리자를 추가할 때 해당 이벤트의 이벤트 처리자가 이미 등록되었는지 확인하는 조건문을 추가해 이벤트 처리자를 덮어쓰지 않도록 수정하는 경우도 조건문 추가를 통해 수정한 예제다.

함수호출 조정을 통해 20 개의 오류를 수정했다. 함수호출 조정은 함수호출의 추가, 제거 및 그 순서의 변경을 포함한다. 또한 본 연구에서는 함수의 이름을 바꾸어 외부 라이브러리 등의 함수 이름과 중복되는 것을 피하는 방법으로 오류를 수정한 경우도 함수호출 변경으로 보았다. 이 방법은 주로 이벤트 처리자와 관련된 오류를 수정할 때 사용되었다. 또한 데이터베이스 등 외부 요소를 사용할 때 외부 요소의 초기화 및 연결 함수 등의 순서 조정을 통해 수정하는 경우도 많았다.

변수의 변화를 통해서는 38 개의 오류를 수정했다. 변수의 변화 방법은 함수의 parameter, 변수 등의 값을 변경한 모든 종류의 수정을 포함한다. 이 방법은 주로 외부 요소와 관련된 오류와 기타 오류 중 프로그래밍 실수로 인한 오류를 수정할 때 사용되었다.

기타 수정 방법으로는 문제가 되는 함수 등을 완전히 다시 구현하는 방법, 특정 기능의 삭제 등이 있었다. 이러한 방법들은 거의 모든 오류의 원인에 대해 적용되었고, 간단한 해결책이 없을 때 사용되었다.

8. 그 밖의 오류 특성

8.1 결함의 위치

자바스크립트와 관련된 오류만을 대상으로 조사했음에도 불구하고, 결함의 위치는 자바스크립트 코드상에만 존재하지 않고 HTML 문서상에도 존재했다. 조사한 오류 100 개 중 17 개의 오류는 HTML 문서에도 결함이 있었다. 또한 이 중 1 개의 오류는 자바스크립트 코드상에 결함이 존재하지 않고, HTML 문서상에만 결함이 존재했다. HTML 문서상에 자바스크립트 관련 결함이 공존하는 이유는 이벤트 처리자 등록이 HTML 문서에서도 이루어지기 때문이다. 실제로 HTML 문서상에 결함이 존재하는 17 개의 오류는 모두 이벤트 처리자와 관련된 오류다. 이는 HTML 문서 작성시 연관되는 자바스크립트 함수 등을 명확히 해야 한다는 것을 의미한다.

8.2 오류 수정의 복잡도

오류 수정은 복잡하지 않았다. 100 개의 오류 중 51 개의 오류는 각각의 오류 수정을 위해 변경된 코드가 20 줄이 넘지 않았다. 또한 각각의 오류 수정을 위해 변경된 코드는 평균 약 32 줄이고, 중간값은 19 줄이다. 이는 자바스크립트 오류는 보통 간단한 실수로 인해 초래된다는 점을 시사한다.

8.3 오류의 발생 환경

어떤 오류는 특정 환경에서만 발생하였다. 특정 환경에는 사용하는 운영체제, 자바스크립트 엔진(웹 브라우저) 등이 포함된다. 본 연구에서 조사한 오류 100 개 중 22 개는 특정 환경에서만 발생하였고, 33 개의 오류가 실행 환경과 무관하게 발생하였다. 나머지 45 개의 오류는 다양한 실행환경에서 오류가 발생하지만, 오류가 발생하지 않는 실행환경도 존재하는 경우다. 특히 성능이나 레이아웃과 관련된 오류는 특정 자바스크립트 엔진에 한해 발생하는 경향을 보인다.

9. 자바스크립트 오류 분석 결과의 활용방안

9.1 개발자 측면 활용방안

본 연구에서 조사한 오류의 43%는 자료의 부재로 인해 발생했다. 이는 개발자가 클라이언트 사이드에서의 자료 접근에 대해 고려해야 한다는 것을 의미한다. 또한 웹 상의 자료에 접근할 경우에도 일시적인 네트워크 장애 등을 고려해 신뢰성있는 코드 구현을 해야 클라이언트 사이드에서 오류 없이 어플리케이션을 사용할 수 있을 것이다.

본 연구에서 조사한 오류의 15%는 단순한 프로그래밍 실수로 인한 것이었다.. 따라서 개발자는 개발 단계에서 간단한 실수를 통해 오류를 만들지 않기 위해 더욱 노력을 해야 한다.

14%의 오류는 외부 요소의 잘못된 사용으로 인해 발생했다. 웹 상에 있는 외부 요소의 경우는 네트워크만 고려하면 되겠지만, 클라이언트 사이드의 어플리케이션을 사용하는 경우에는 해당 외부 어플리케이션의 버전이나 호환성 등 다양한 경우를 고려해서 구현해야 한다. 또한 외부 자바스크립트 라이브러리를 사용할 때, 변수 및 함수 이름 사용에 주의를 기울여야 한다. 자바스크립트의 변수 이름은 모든 영역에서 공통으로 사용되고, 동적으로 관리되기 때문에 세심한 주의가 필요하다. 특히 외부 라이브러리에서 사용하는 변수 및 함수 이름을 반복해서 사용해 잘못된 변수 및 함수를 사용하는 일이 없도록 해야 한다.

9.2 테스터 측면 활용방안

테스터는 다양한 클라이언트 사이드 환경을 고려해 테스트해야 한다. 보고된 오류 중 상당수는 특정 환경 (e.g., 운영체제, 자바스크립트 엔진)에서만 재현되었다. 따라서 테스트 단계에서 다양한 사용자 환경을 고려해 테스트해야 보다 많은 오류를 서비스 전에 찾아 수정할 수 있을 것이다.

또한 테스터는 자바스크립트의 유연성에 대해서도 고려해야 한다. 특히 자바스크립트는 동적 타입 시스템을 사용하기 때문에 변수에 값을 읽거나 쓸 때 변수의 타입이 달라도 오류 없이 실행되는 경우가 많다. 하지만 이는 추후에 예기치 못한 부분에서 오류를 발생시킬 수 있기 때문에 자바스크립트의 유연성을 고려해, 테스트 시 매 단계마다 변수의 타입을 확인하는 등의 추가적인 노력이 필요하다. 또한 이러한 자바스크립트의 유연성은 일반적으로 자바스크립트 엔진에 따라 다른 양상을 보이기 때문에 다양한 자바스크립트 엔진을 활용해 테스트 하는 것이 필요하다.

10. 결론 및 추후 연구 과제

본 연구에서는 실제 자바스크립트 어플리케이션의 오류를 조사하고 분류했다. 또한 조사한 오류를 바탕으로 개발자와 테스터에게 활용방안을 제안하고 있다. 본 연구의 주요 기여는 다음과 같다.

- 최초의 실제 자바스크립트 기반 웹 어플리케이션을 대상으로 한 오류의 영향, 원인, 수정 방법 등을 포함한 오류 조사
- 실제 자바스크립트 기반 웹 어플리케이션의 오류를 그 영향, 원인 수정 방법을 기준으로 분류
- 자바스크립트 개발자와 테스터에게 활용방안 제공

추후 연구 과제는 크게 두 가지 관점에서 생각할 수 있다. 하나의 관점은 본 연구에서 조사한 오류를 보다 심층적으로 분석하고, 기존의 연구와 연계해 자바스크립트의 동적, 정적 특성과의 관계를 파악하는 것이다. 또한 본 연구에서는 오류의 영향과 원인, 수정 방법에 대한 구체적인 상관관계를 분석하지 않았지만 이를 분석하는 것 또한 필요하다.

또 다른 관점은 조사한 오류의 특성을 활용해 자바스크립트 기반 웹 어플리케이션의 특성에 맞춘 테스트 기법을 개발하는 것이다.

참고문헌

[1] H. Bidgoli, The Internet Encyclopedia. John Wiley & Sons Inc, 2004.

[2] ECMA International. ECMA-262: ECMAScript Language Specification. ECMA (European Association for Standardizing Information and Communication Systems), Geneva, Switzerland, third edition, December 1999.

[3] S. Artzi, J. Dolby, S. H. Jensen, A. Møller, et al., "A framework for automated testing of javascript web applications," presented at the Proceedings of the 33rd International Conference on Software Engineering, Waikiki, Honolulu, HI, USA, 2011.

[4] A. Mesbah, E. Bozdag, and A. v. Deursen, "Crawling AJAX by Inferring User Interface State Changes," presented at the Proceedings of the 2008 Eighth International Conference on Web Engineering, 2008.

[5] A. Mesbah and A. v. Deursen, "Invariant-based automatic testing of AJAX user interfaces," presented at the Proceedings of the 31st International Conference on Software Engineering, 2009

[6] P. Saxena, D. Akhawe, S. Hanna, F. Mao, S. McCamant, and D. Song, "A Symbolic Execution Framework for JavaScript," presented at the Proceedings of the 2010 IEEE Symposium on Security and Privacy, 2010.

[7] F. S. Ocariza, K. Pattabiraman, and B. Zorn, "JavaScript Errors in the Wild: An Empirical Study," in Software Reliability Engineering (ISSRE), 2011

IEEE 22nd International Symposium on, 2011, pp. 100-109.

[8] P. Ratanaworabhan, B. Livshits, and B. G. Zorn, "JSMeter: comparing the behavior of JavaScript benchmarks with real web applications," presented at the Proceedings of the 2010 USENIX conference on Web application development, Boston, MA, 2010.

[9] G. Richards, S. Lebresne, B. Burg, and J. Vitek, "An analysis of the dynamic behavior of JavaScript programs," presented at the Proceedings of the 2010 ACM SIGPLAN conference on Programming language design and implementation, Toronto, Ontario, Canada, 2010.

[10] C. Yue and H. Wang, "Characterizing insecure JavaScript practices on the web,"in Intl. Conference on World Wide Web (WWW), 2009, pp. 961-970.

[11] D. Jang, R. Jhala, S. Lerner, and H. Shacham, "An empirical study of privacy-violating information flows in JavaScript web applications," in ACM Conference on Computer and Communications Security, 2010, pp. 270-283.

[12] T. Mikkonen and A. Taivalsaari, "Using JavaScript as a Real Programming Language" Sun Microsystems Laboratories Technical Report, vol. 168, 2007.

[13] Alexa, The Web Information Compony, www.alexa.com

[14] G. Jin, L. Song, X. Shi, J. Scherpelz, and S. Lu, "Understanding and detecting real-world performance bugs," presented at the Proceedings of the 33rd ACM SIGPLAN conference on Programming Language Design and Implementation, Beijing, China, 2012.