DeloreanJS: Un Debugger en el Tiempo para JavaScript

Paul Leger
Escuela de Ingeniería
Universidad Católica del Norte
Coquimbo, Chile
pleger@ucn.cl

AAAA BBBB
Escuela de Ingeniería
Universidad Católica del Norte
Coquimbo, Chile
aaaa.bbbb@alumnos.ucn.cl

XXXX YYYY
Escuela de Ingeniería
Universidad Católica del Norte
Coquimbo, Chile
xxxx.yyyy@alumnos.ucn.cl

***Resumen*—Aplicaciones Web, usando JavaScript, son desarrolladas cada vez con mayor frecuencia. Como en la mayoría de los entornos de desarrollos, una aplicación Web puede adquirir defectos de software (conocido como *bugs*) cuyos síntomas se aprecian durante este desarrollo e incluso, siendo peor, en su puesta en producción. Por ello, el uso de debuggers son sumamente útiles para detectar estos bugs. Lamentablemente, los actuales debuggers solamente avanzan hacia adelante en la ejecución para detectar el bug y no permiten retornar hacia un punto anterior para tomar acciones asociadas al bug detectado. Por ejemplo, probar si el mismo bug podría gatillarse con otro valor de una variable. Usando el concepto de continuaciones, este artículo presenta un debugger para JavaScript que permite devolverse en el tiempo con el fin que un programador pueda volver a verificar y probar el contexto alrededor de un bug. Este debugger, llamado DeloreanJS, ha sido implementado como una extensión para Mozilla Firefox con el fin que la comunidad de desarrolladores puedan usarlo.**

***Index Terms*—DeloreanJS, JavaScript, Debugger, Web application, Continuations**

I. Introducción

La industria del software se mueve fuertemente hacia el desarrollo de aplicaciones Web, siendo testigo un gran número de migraciones de aplicaciones *standalone* a la Web; los ejemplos van desde convertir un documento Word a un formato PDF [1] hasta un sistema ERP (*Enterprise Resource Planning*) [2]. Para construir estas aplicaciones, uno de los lenguajes más usados en JavaScript, cuya presencia en el entorno de la Web es alrededor del 95 % [3]. Por ello, cada vez estas aplicaciones se han vuelto más complejas y con mayor riesgo de introducir defectos de software (*a.k.a. bugs*).

Detectar y reparar bugs representa una de las tareas más costosa en el proceso de desarrollo de software, y las aplicaciones Web no son la excepción. Para ello, un conjunto de debuggers se han propuestos, desde el que incluye un navegador en su distribución (*e.g.,* Mozilla Firefox developer tools) hasta omniscientes debuggers que pueden recorrer a través de la ejecución para encontrar el origen de un bug [4], [5]. Lamentablemente, la gran parte de estos debuggers son *post-morter*, y por ende, solo permiten mostrar la ocurrencia de un bug sin entregar la posibilidad de retroceder *en el tiempo* para entregar la posibilidad de repararlo mientras se ejecuta el programa o manipular el estado de un programa alrededor del bug con el fin de mejorar su comprensión.

Este artículo científico presenta DeloreanJS, un debugger que permite al desarrollador fijar *timepoints* (en cambio de *breakpoints*) en una aplicación en JavaScript con el fin de permitir al programador retornar en el tiempo a esos puntos y modificar valores de variables para continuar la ejecución del programa. Esta novedosa visión de DeloreanJS, la cual está disponible como una extensión de Mozilla Firefox, permite:

1. Modificar los valores de las variables asociado a un bug encontrado para mejorar la comprensión de este bug. Esto potencialmente ahorra un gran número de ejecuciones en un contexto similar para descubrir la verdadera razón del bug.
2. Probar escenarios hipotéticos de ejecución de una aplicación usando un timepoint de DeloreanJS. Esto permite descubrir diversas evoluciones de la aplicación cambiando valores de algunas variables.
3. item Mantener una aplicación funcionando usando los timepoints para potencialmente reparar un bug durante la ejecución. Lo anterior significa que no es siempre necesario detener la ejecución de la aplicación con el fin de realizar un analísis post-morter. Esto es importante porque detener una aplicación significa perder el contexto de ejecución asociado al bug.

Para construir DeloreanJS, utilizamos la abstracción *continuación* [6] de los lenguajes de programación funcional como Scheme [7]. En los lenguajes funcionales, una continuación permite al programador capturar y guardar, como valor de primera clase, un momento de ejecución (*program counter* y *stack*) de una aplicación. Extendiendo esta abstracción para lenguajes no funcionales como JavaScript, nosotros habilitamos a DeloreanJS para que ofrezca la posibilidad a un programador de usar un método que agregue timepoints al programa, y así poder volver esos momentos de ejecución y modificarlos cuando el programador lo requiera.

El artículo está organizado como sigue. La sección II presenta diferentes ejemplos de aplicación de DeloreanJS, donde se puede apreciar la interfaz gráfica de nuestra propuesta. Luego, se presenta DeloreanJS, detallando cómo se usa el concepto de continuaciones. En la sección IV se discuten herramientas similares. Finalmente, sección V concluye y describe lineamientos sobre el trabajo futuro de nuestra propuesta.

Disponibilidad. El código fuente de la implementación de DeloreanJS se encuentra en `http://github.com/fruizrob/delorean` y su compilada versión se encuentra como extensión en Mozilla Firefox en `http://xxx.com`.

II. Un Tour por DeloreanJS

PL *Cada ejemplo debe tener el nombre de lo que hace* □

II-A. Ejemplo 1

PL *Explicar con imagenes el ejemplo 1* □

II-B. Ejemplo 2

PL *Explicar con imagenes el ejemplo 2* □

II-C. Ejemplo 3

PL *Explicar con imagenes el ejemplo 3* □

III. DeloreanJS

III-A. Continuaciones

PL *Explicar continuaciones [6]* □

III-B. Capturando el Estado

PL *Explicar su forma de capturar estados* □

III-C. Extensión en el Navegador

PL *Explicar la aplicación en Mozilla Firefox* □

IV. Trabajo Relacionado

PL *Aqui se debe comparar con tres herramientas similares a DeloreanJS* □

Herramienta 1.

Herramienta 2.

Herramienta 3.

V. Conclusiones y Trabajo Futuro

PL *TODO* □

Trabajo Futuro.

Referencias

[1] "Smallpdf: Word to pdf," 2019. [Online]. Available: https://smallpdf.com/word-to-pdf

[2] "Oracle erp cloud," 2019. [Online]. Available: http://www.oracle.com/ERP

[3] "Usage of client-side programming languages," https://w3techs.com/technologies/history_overview/client_side_language/all, accessed: 2019-03-14.

[4] Z. Azar, "Peccit:an omniscient debugger for web development," Master's thesis, University of Denver, Denver, United States, 2016.

[5] E. Barr, M. Marron, E. Maurer, D. Moseley, , and G. Seth, "Time-travel debugging for javascript/node.js," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Seattle, WA, USA, Nov. 2016, pp. 1003–1007.

[6] D. P. Friedman and M. Wand, "Reification: Reflection without metaphysics," in *Proceedings of the Annual ACM Symposium on Lisp and Functional Programming*, Aug. 1984, pp. 348–355.

[7] R. A. Kesley and J. A. Rees, "A tractable Scheme implementation," *Lisp and Symbolic Computation*, vol. 7, no. 4, pp. 315–335, 1995.