

Nexus – Elemento: PDF Viewer



¿Qué es?

PDF Viewer es un componente de tipo **Display**. Muestra un visor embebido de archivos PDF, con navegación entre páginas. Al arrastrarlo al lienzo viene con un PDF de ejemplo precargado (documento de 14 páginas).

1 / 14

PrevNext

Trace-based Just-in-Time Type Specialization for Dynamic Languages

Andreas Gal¹, Brendan Eich², Mike Shaver³, David Anderson⁴, David Mandelin⁵,
Mohammad R. Haghighat⁶, Blake Kaplan⁷, Graydon Hoare⁸, Boris Zharsky⁹, Jason Orendorff¹⁰,
Jesse Ruderman¹¹, Edwin Smith¹², Rick Reitmaier¹³, Michael Bebenita¹⁴, Mason Chang¹⁵, Michael Franz¹⁶

Mozilla Corporation^{*}
[gal, brendan, shaver, danderson, dmandelin, mrbkap, graydon, bz, jorendorff, jruderman]@mozilla.com

Adobe Corporation²
[edsmith, roitman]@adobe.com

Intel Corporation³
[mohammad.r.haghighat]@intel.com

University of California, Irvine⁴
[abebenit, changa, franz]@uci.edu

Abstract

Dynamic languages such as JavaScript are more difficult to compile than statically typed ones. Since no concrete type information is available, traditional compilers need to emit generic code that can handle all possible type combinations at runtime. We present an alternative compilation technique for dynamically-typed languages that identifies frequently executed loop traces at run time and then generates machine code on the fly that is specialized for the actual dynamic types occurring on each path through the loop. Our method provides cheap inter-procedural type specialization, and an elegant and efficient way of incrementally compiling lazily discovered alternative paths through nested loops. We have implemented a dynamic compiler for JavaScript based on our technique and we have measured speedups of 10x and more for certain benchmark programs.

Categories and Subject Descriptors: D.3.4 [Programming Languages]: Processors — Incremental compilers, code generation.

General Terms: Design, Experimentation, Measurement, Performance.

Keywords: JavaScript, just-in-time compilation, trace trace.

1. Introduction

Dynamic languages such as JavaScript, Python, and Ruby, are popular since they are expressive, accessible to non-experts, and make deployment as easy as distributing a source file. They are used for small scripts as well as for complex applications. JavaScript, for example, is the de facto standard for client-side web programming and is used for the application logic of browser-based productivity applications such as Google Mail, Google Docs and Zimbra Collaboration Suite. In this domain, in order to provide a fluid user experience and enable a new generation of applications, virtual machines must provide a low startup time and high performance.

Compilers for statically typed languages rely on type information to generate efficient machine code. In a dynamically typed programming language such as JavaScript, the types of expressions may vary at runtime. This means that the compiler can no longer easily transform operations into machine instructions that operate on one specific type. Without exact type information, the compiler must emit slower generalized machine code that can deal with all potential type combinations. While compile-time static type inference might be able to gather type information to generate optimized machine code, traditional static analysis is very expensive and hence not well suited for the highly interactive environment of a web browser.

We present a trace-based compilation technique for dynamic languages that reconciles speed of compilation with excellent performance of the generated machine code. Our system uses a mixed-mode execution approach: the system starts running JavaScript in a fast-starting bytecode interpreter. As the program runs, the system identifies hot (frequently executed) bytecode sequences, records them, and compiles them to fast native code. We call such a sequence of instructions a *trace*.

Unlike method-based dynamic compilers, our dynamic compiler operates at the granularity of individual loops. This design choice is based on the expectation that programs spend most of their time in hot loops. Even in dynamically typed languages, we expect hot loops to be mostly type-stable, meaning that the types of values are invariant. (17) For example, we would expect loop counters that start as integers to remain integers for all iterations. When both of these expectations hold, a trace-based compiler can cover the program execution with a small number of type-specialized, efficiently compiled traces.

Each compiled trace covers one path through the program with one mapping of values to types. When the VM executes a compiled trace, it cannot guarantee that the same path will be followed or that the same types will occur in subsequent loop iterations.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PLDI'09, June 15–20, 2009, Dublin, Ireland.
Copyright © 2009 ACM 978-1-60558-109-6...\$5.00

Propiedades

Al seleccionar PDF Viewer en el lienzo, el panel derecho muestra **“Properties of PDFViewer”** con las siguientes opciones:

- **Component ID:** identificador único del componente, autogenerado (ej. pdf_115c8c3d).
- **Visibility → Hidden on load:** casilla para que el componente esté oculto al cargar la pantalla. Puede mostrarse luego mediante una acción de tipo **Show** (ver Events & Actions).
- **PDF URL:** campo donde se coloca la URL del archivo PDF a mostrar.
- **Initial Page:** número de página con la que se abre el documento por defecto (ej. 1).
- **Scale (0.5 – 3):** nivel de zoom/escala con el que se muestra el PDF, en un rango de 0.5 a 3 (valor por defecto: 1).
- **Behavior → Show Toolbar:** casilla que muestra u oculta la barra de herramientas del visor (indicador de página, botones Prev/Next).
- **Style → Custom Class:** campo para asignar una clase CSS personalizada.
- **Style → Custom Style:** campo de texto para CSS personalizado aplicado directamente a este componente.
- **Delete Component:** elimina el componente del lienzo.
- **Events & Actions:** sistema para definir comportamientos interactivos. Ver documentación aparte: [Nexus – Events & Actions](#) .

The screenshot displays the Nexus IDE interface. The main workspace shows a PDF document titled "Trace-based Just-in-Time Type Specialization for Dynamic Languages". The document content includes an abstract, introduction, and a conclusion. The right sidebar, titled "Properties of PDFViewer", contains the following settings:

- Component ID:** pdf_115c8c3d
- Visibility:** ☐ Hidden on load
- PDF URL:** <https://mozilla.github.io/pdf.js/web/cc>
- Initial Page:** 1
- Scale (0.5 - 3):** 1
- Behavior:** ☒ Show Toolbar
- Style:**
 - Custom Class:**
 - Custom Style:**
- Delete Component:** (button)
- Events & Actions:** [+ Add Action](#)

Ejemplo de uso

Para mostrar un PDF propio, se reemplaza **PDF URL** por la dirección del archivo (debe ser una URL pública accesible, terminada en .pdf). Con **Initial Page** se puede hacer que el documento se abra directamente en una página específica en vez de la primera, y con **Scale** se ajusta el zoom inicial del documento.