pdf.js

Ionuț G. Stan - wurbe #48

pdf.js

- what
- why
- how
- demo

pdf.js

- what
- why
- how
- demo

pdf.js

- PDF renderer written in JavaScript

pdf.js

- PDF renderer written in JavaScript
- specification partially implemented

pdf.js

- PDF renderer written in JavaScript
- specification partially implemented
- good rendering quality

pdf.js

- PDF renderer written in JavaScript
- specification partially implemented
- good rendering quality
- supports text selection

pdf.js

- PDF renderer written in JavaScript
- specification partially implemented
- good rendering quality
- supports text selection
- experimental Firefox extension

pdf.js

- PDF renderer written in JavaScript
- specification partially implemented
- good rendering quality
- supports text selection
- experimental Firefox extension
- original creator: Andreas Gal

pdf.js

- PDF renderer written in JavaScript
- specification partially implemented
- good rendering quality
- supports text selection
- experimental Firefox extension
- original creator: Andreas Gal
- now maintained by Mozilla

pdf.js

- what
- why
- how
- demo

pdf.js

- having fun

`pdf.js`

- having fun probably

`pdf.js`

- having fun probably
- but I use it in production

pdf.js

- having fun probably
- but I use it in production
- we're replacing a Flash viewer

pdf.js

- having fun probably
- but I use it in production
- we're replacing a Flash viewer
- contributed a few fixes

mozilla / **pdf.js**

Code | Network | Pull Requests 6 | Issues 146

Files | **Commits** | Branches 4 | Tags 2 | Downloads

pdf.js / Commit History

Nov 02, 2011

Fix same origin policy issue when adding @font-face rules
igstan authored November 02, 2011

Oct 28, 2011

Merge branch 'master' of https://github.com/andreasgal/pdf.js
igstan authored October 29, 2011

Fix lint errors
igstan authored October 29, 2011

Add eq test for close path rendering bug
igstan authored October 28, 2011

Merge branch 'master' of https://github.com/andreasgal/pdf.js
igstan authored October 28, 2011

Log error stacktrace only when available
igstan authored October 28, 2011

pdf.js

- what
- why
- how
- demo

pdf.js

- rendering: currently `canvas`

`pdf.js`

- rendering: currently `canvas`
- might use SVG in the future

`pdf.js`

- rendering: currently `canvas`
- might use SVG in the future
- parsing: JS typed arrays (ArrayBuffer)

`pdf.js`

- rendering: currently `canvas`
- might use SVG in the future
- parsing: JS typed arrays (ArrayBuffer)
- XHR2 for fetching PDFs

`pdf.js`

- rendering: currently `canvas`
- might use SVG in the future
- parsing: JS typed arrays (ArrayBuffer)
- XHR2 for fetching PDFs
- embedded fonts: web fonts in data URIs

pdf.js

- what
- why
- how
- demo

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

Andreas Gal*+, Brendan Eich*, Mike Shaver*, David Anderson*, David Mandelin*,
Mohammad R. Haghighat$, Blake Kaplan*, Graydon Hoare*, Boris Zbarsky*, Jason Orendorff*,
Jesse Ruderman*, Edwin Smith#, Rick Reitmaier#, Michael Bebenita+, Mason Chang+#, Michael Franz+

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

Adobe Corporation#
{edwsmith,rreitmai}@adobe.com

Intel Corporation$
{mohammad.r.haghighat}@intel.com

University of California, Irvine+
{mbebenit,changm,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 trees.

1. Introduction

Dynamic languages such as JavaScript, Python, and Ruby, are popular 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*.

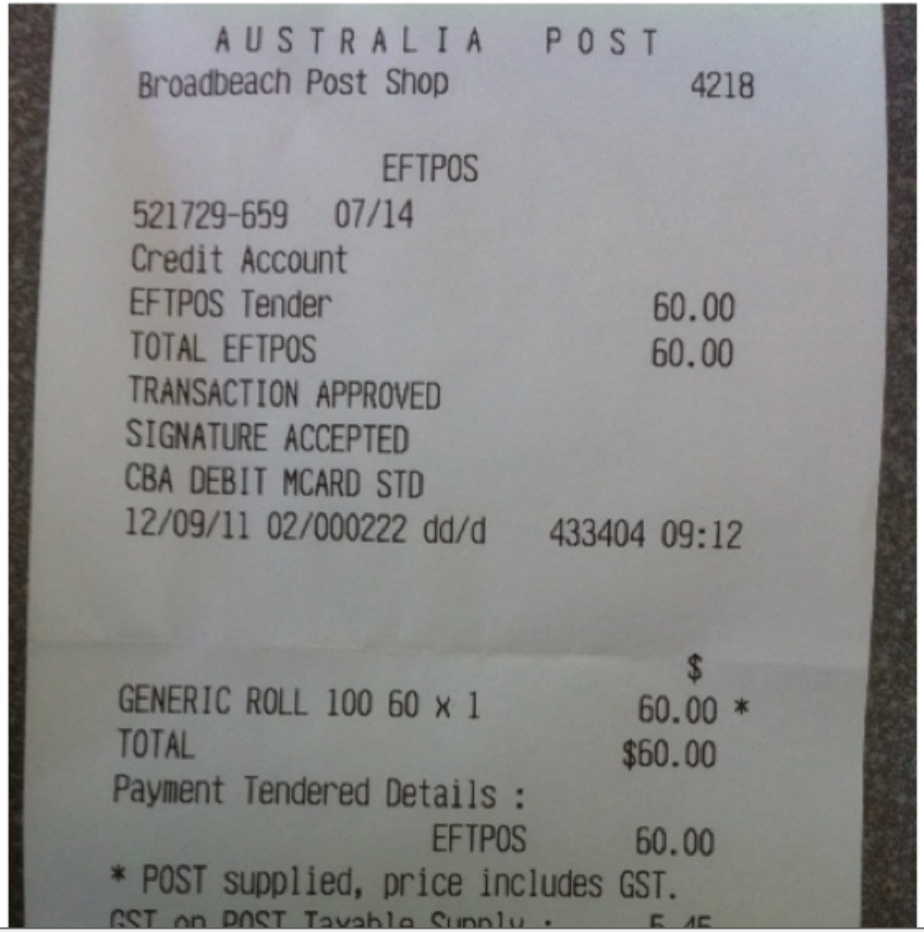
pdfs/receipt-001.pdf

pdfs/receipt-001.pdf

file:///Users/igstan/Projects/clients/shoeboxed/sbx-pdf-viewer/vie

MDN Search

page 1 of 1



AUSTRALIA POST
Broadbeach Post Shop 4218

EFTPOS
521729-659 07/14
Credit Account
EFTPOS Tender 60.00
TOTAL EFTPOS 60.00
TRANSACTION APPROVED
SIGNATURE ACCEPTED
CBA DEBIT MCARD STD
12/09/11 02/000222 dd/d 433404 09:12

$
GENERIC ROLL 100 60 x 1 60.00 *
TOTAL $60.00
Payment Tendered Details :
EFTPOS 60.00
* POST supplied, price includes GST.
GST on POST Taxable Supply : 5.45

thanks

- https://github.com/mozilla/pdf.js
- http://igstan.ro