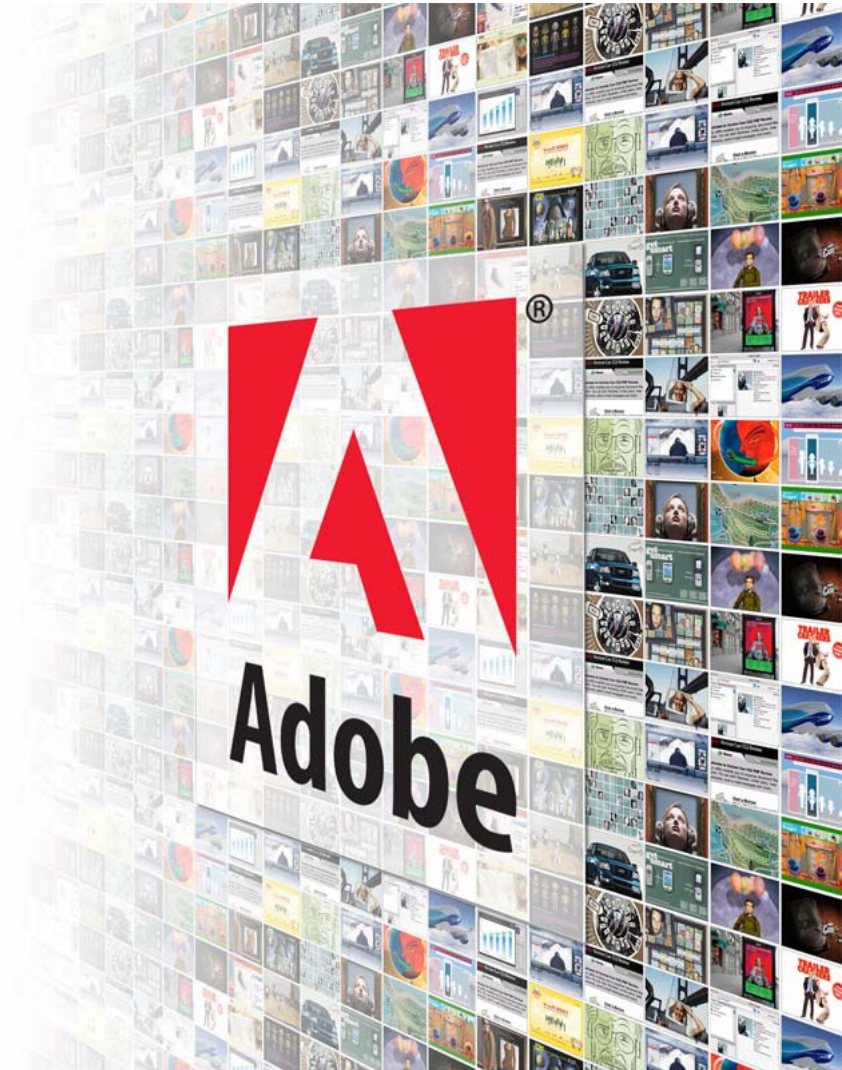


# Adobe® Flash® Player ActionScript Virtual Machine (Tamarin)

Rick Reitmaier  
Sr. Scientist  
Adobe Systems

December 6, 2006



# Adobe Today

## Worldwide Offices



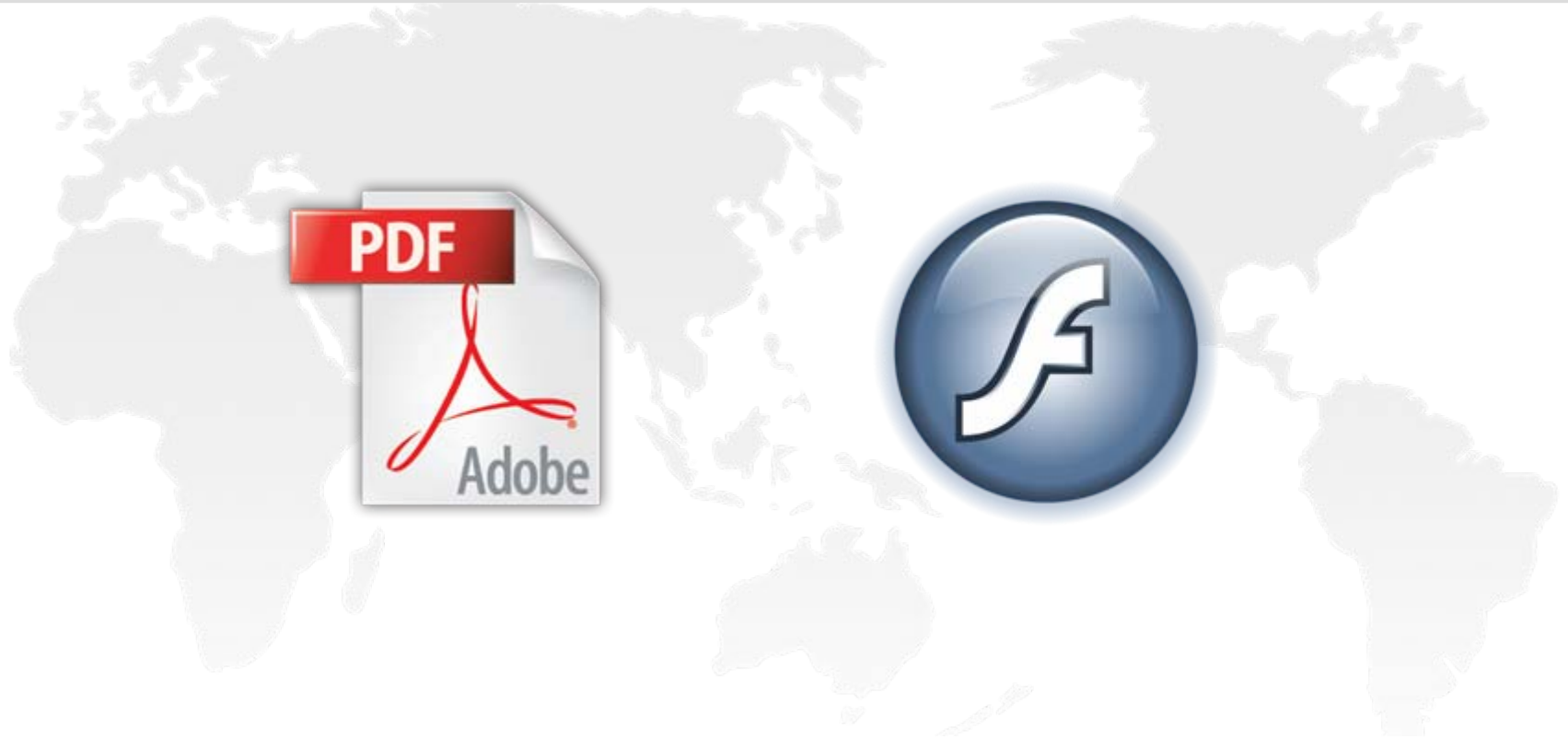
## Corporate Headquarters – San Jose, California



## Key Statistics

Adobe FY 2005 Revenue	\$1.966B
Macromedia FY 2005 Revenue	\$436M
Years in Business	23
Employees	5,000

# Widest Reach in the World



**600 million**  
PCs and devices



# Device shipment growth is explosive



- 150+ handset models
- 300+ device models
- Over 100 million devices shipped

FUJITSU

HITACHI

KYOCERA

MITSUBISHI

SAMSUNG

NEC

NOKIA

SANYO

SHARP

LG

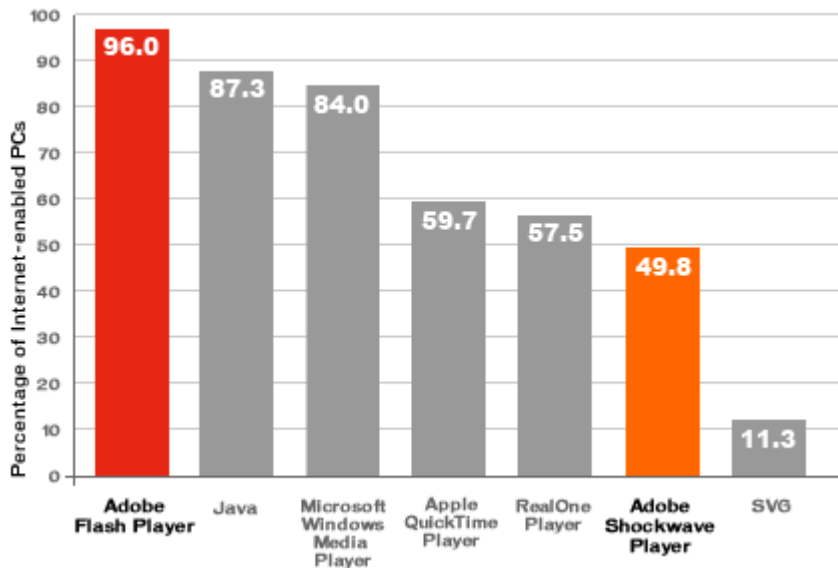
Sony Ericsson

TOSHIBA

SIEMENS

MOTOROLA

# Plug-In Technology Penetration: Mature Markets



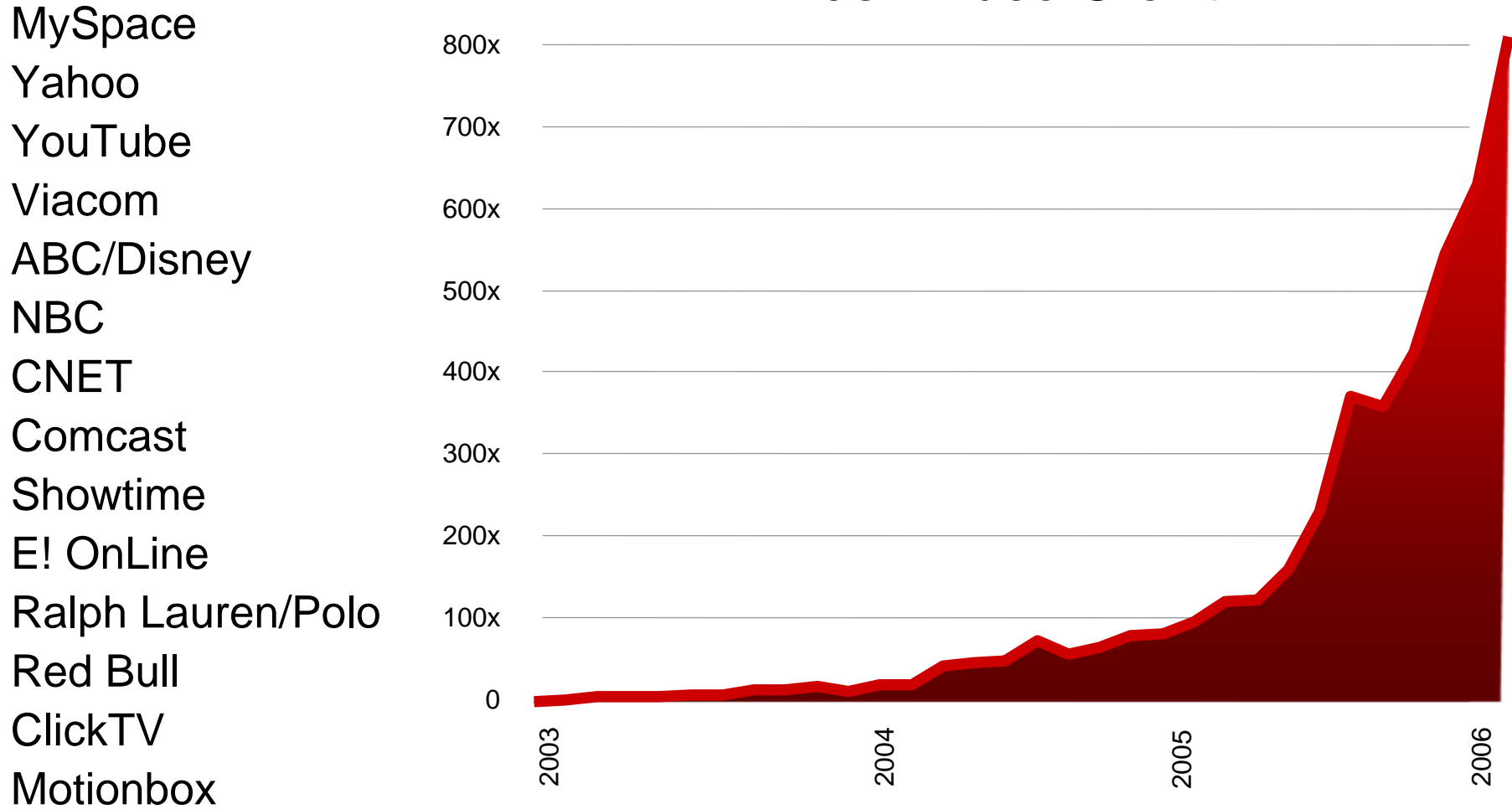
## Internet Connected Desktops

- Flash Player typically reaches 80% adoption in < 1yr each release.
- Version 8 (Fall 2005) achieved a higher adoption in less time.
- Version 9 (Summer 2006) appears to be following same curve as version 8.

\* MillwardBrown, September 2006

# Flash Video Momentum

## Flash Video Growth



# Anatomy of a Rich Internet Application

Menus & Navigation Controls

Real Time Data Push & Alerting

Mapping, Geo-Coding, Annotations & White boarding

Bi-Directional Audio & Video

Data Visualization & Collaboration

Resizable Views/Effects/Transitions

Rich Data Entry

Chat

Data Synchronization & Conflict Resolution

Offline

The screenshot shows a web application interface with several key components:

- Navigation Menu:** HOME, EMAIL, CONTACTS, MY GALLERY, VIDEOS, WEB STUFF, HELP & SUPPORT, PERSONALISE, LOG OUT
- Portfolio Table:**

Symbol	Name	Open	Last	Change	High	Low
ADBE	Adobe Systems Incorporated	39.20	39.32	-.12	40.57	39.11
	American International Group Inc	62.92	64.14	.11	64.04	61.21
	Boeing Company	83.45	81.41	.15	84.28	79.72
	Citigroup Inc	48.18	47.79	-.16	49.92	47.59
	Coca-Cola Bottling Co. Consolidated	48.20	49.80	-.18	51.05	47.24
	Conocophillips	67.14	66.75	-.09	69.40	64.79
	Corp New	58.95	60.60	.23	61.30	58.21
		33.61	34.88	.08	35.07	33.21
	Genzyme Corporation	61.16	63.49	-.18	63.88	59.87
GM	General Motors Corporation	19.80	19.79	-.04	20.63	19.56
GOOG	Google Inc	417.93	451.12	.58	457.21	417.02
IBM	International Business Machines Co	82.34	79.94	-.14	84.40	79.81
JBLU	JetBlue Airways Corporation	10.57	11.00	.00	11.21	10.45
MCD	McDonald's Corporation	34.57	34.45	-.04	35.58	33.82
MOT	Motorola, Inc.	21.35	21.28	-.09	21.62	20.74
SAP	SAP AG	54.63	54.92	-.03	56.27	54.22
VZ	Verizon Communications	33.03	31.42	-.15	34.17	31.54
	Walmart Stores	45.62	44.80	-.14	46.75	44.51
	Mobile Corp	61.56	59.59	-.07	63.24	59.63
- Regional Breakdown [Jan-04]:** A pie chart showing regional data with labels for Japan and North America. A video call window is overlaid on the chart.
- Map:** A satellite map view showing a city street grid with numbered annotations (1-23) and a 'Bank of America - ATM' marker.
- Form:** A data entry form for 'Company Name: Adobe' with fields for Address (601 Townsend Street), City (San Francisco), State (CA), Zip (25447), and Industry (Computers).
- Data Conflicts:** A section titled 'Data Conflicts' with a message: 'Someone else has just revised the data you are about to update. The differences are:'. It shows a comparison between 'Yours' (601 Townsend Street) and 'Server' (600 Townsend Street) with buttons for 'Use Yours', 'Use Server', and 'Save for Later'.

# All this using a scripting language?

- Strengths
  - low barrier to entry
  - rapid, flexible development
  - familiar
  - pervasive (the web wouldn't compile)
- Opportunities
  - safe, predictable and efficient
  - debugging

# A Tale of Names

## ECMA

- 262 edition 2
- 262 edition 3
- 262 edition 4

## Mozilla

- JavaScript 1.3
- JavaScript 1.4
- JavaScript 1.5
- JavaScript 1.6
- JavaScript 1.7
- JavaScript 2.0

## Adobe

- ActionScript 1.0
- ActionScript 2.0
- ActionScript 3.0

\* JavaScript and ActionScript are trademarked names by Sun Microsystems and Adobe Systems respectively

# A bit of history

- Prototype-based scripting language
  - no classes
  - instances created by cloning other instances
  - Self (Ungar, Smith 1987) influenced design
- Object
  - consists of a collection of properties, also called 'slots'
  - accessing a slot returns a value and may execute code.

# Prototype example

```
var myObj = {};
```

```
myObj.value = "this is string"
```

```
myObj.addProperty("foo", function() { return "foo"; }, null);
```

```
trace(myObj.value); // <= "this is string"
```

```
trace(myObj.foo); // <= "foo"
```

# Objectifying

- Customers requested “Class”
  - added support in Actionscript 2.0
  - stitched together using prototypes
  - pure compiler only change
  - based on unreleased ECMAScript edition 4 proposal
- Explicit data types
  - used as annotation for the compiler to perform error checking
  - ignored when executing

# Object Example

```
class A {  
    var value:String;  
    function get foo():String { return "foo"; }  
}
```

```
var myObj:A = new A();
```

```
myObj.value = "this is string"
```

```
trace(myObj.value);           // <= "this is string"
```

```
trace(myObj.foo);            // <= "foo"
```

# Workflow

- **Compiler**
  - local to developers machine
  - generates bytecode for a stack based machine
- **Bytecode**
  - bytecode is representative of evolution
  - crosses levels of abstraction ; eg. `actionPush` and `actionGetTime`
- **Run-time**
  - simple interpreter

# AVM2

- Trends
  - more code being written
  - complexity of web applications increasing
  - componentization is a growing market
- ActionScript Virtual Machine 2
  - addresses performance
  - maintains backward compatibility; indirectly
  - allows revolutionary innovation to code base

# Flash Player 9



**Flash Player 9**

**AVM1**

**AVM2**

## Enterprise-Class Runtime for Data-Rich Applications

- **New Virtual Machine**
  - Terrific performance improvements
  - Reduced memory consumption
- **ActionScript 3.0**
  - ECMAScript Edition 4 compliant
  - Dynamic and typed programming language
  - External API integrates w/ AJAX & native apps
  - ECMAScript for XML (E4X)
- **Seamless Deployment**
  - One-click upgrade
  - Full backward compatibility

# Challenges

## Constraint

- Flash Player size is ~1MBytes compressed
- amount of bytecode required for 'typical' application must remain small
- startup speed critical for interactive applications

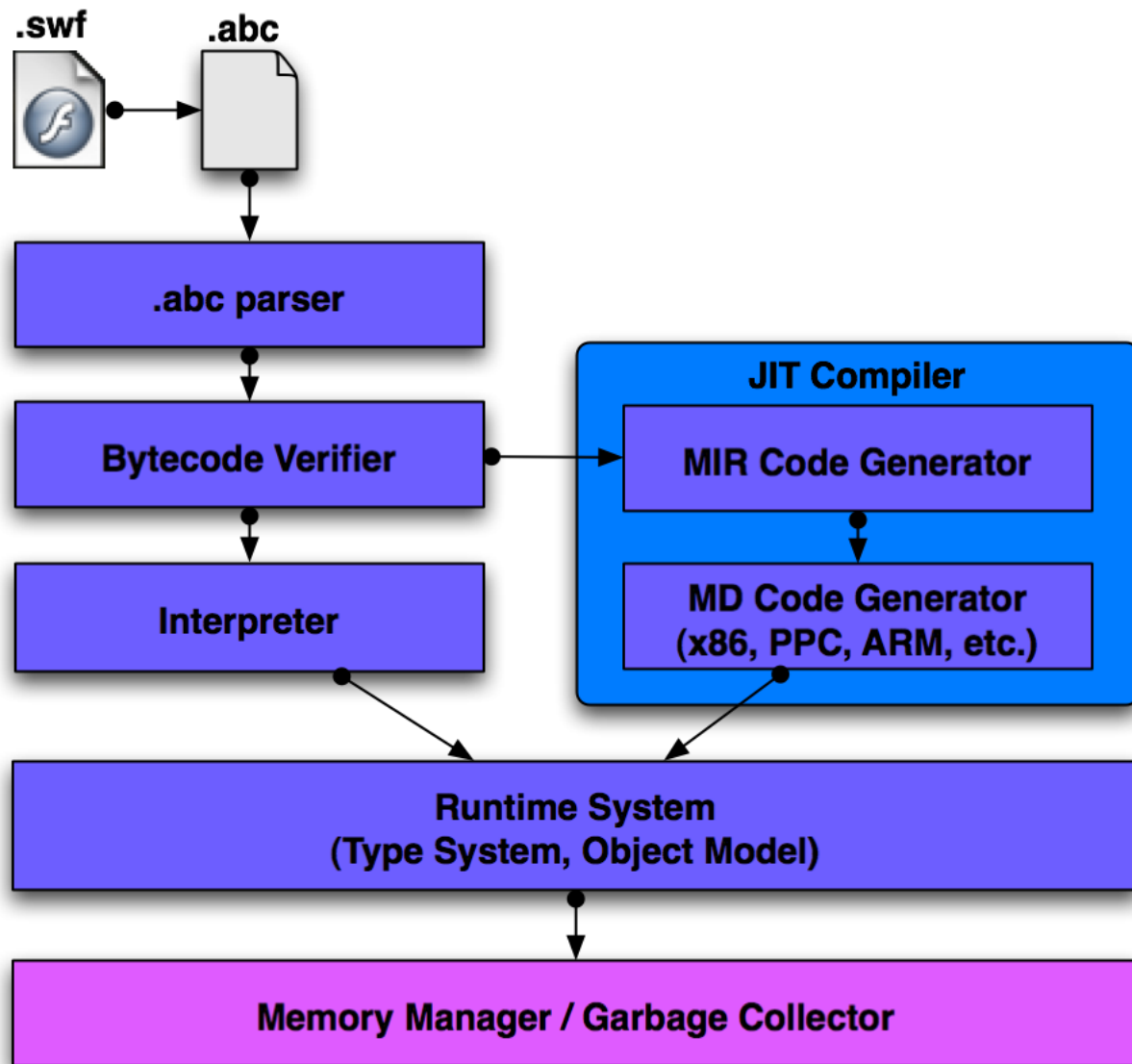
## Solutions

- tight budget for growth
- language specific operations
- variable length encoding
- compression
- novel just-in-time compiler
- novel garbage collector

# AVM2 Design

- Bytecode
  - constant data; strings, numbers, etc
  - type and method descriptors
  - exception tables
  - stack oriented abstract machine
  - object creation, slot access, property search
- Run-time engine
  - separate verification from execution
  - leaner interpreter
  - 2-pass just-in-time compiler
  - revamped garbage collector

# AVM2 Architecture



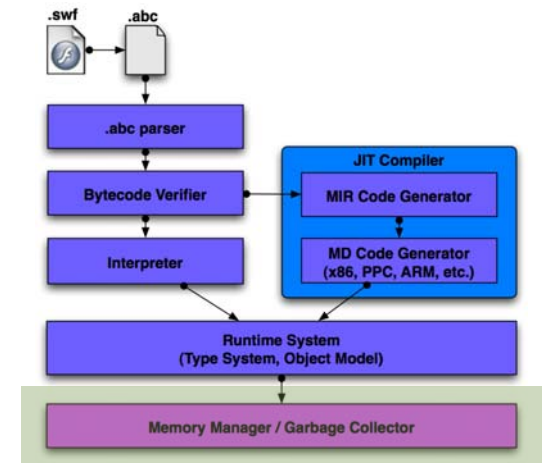
# AVM2 Garbage collector

- Reusable library

- nothing specific to AVM2
- new / delete (unmanaged memory)
- new only (i.e garbage collected)
- debugging aids, profiling

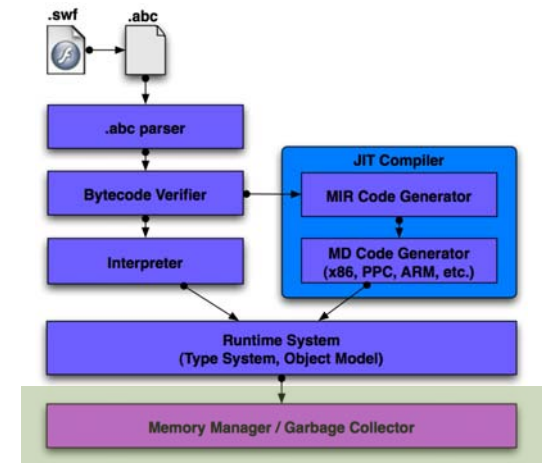
- Characteristics

- deferred reference counting (DRC)
- backed by incremental, conservative mark and sweep
  - incremental = interruptible
  - conservative = random bit pattern in memory may be pointer



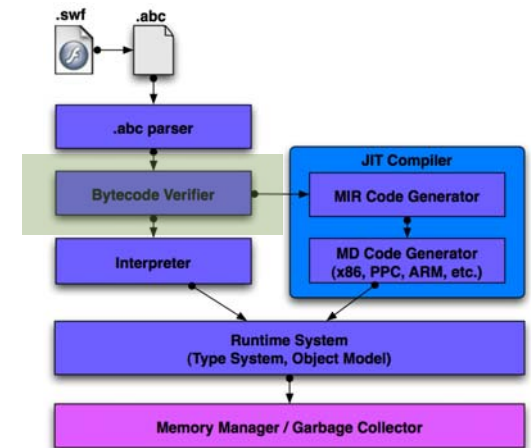
# AVM2 Verifier

- Structural Integrity
  - ensure branches land on valid instructions
  - can't fall off end of code
  - constant references are valid
- Type Safety
  - dataflow analysis to track types
  - early binding



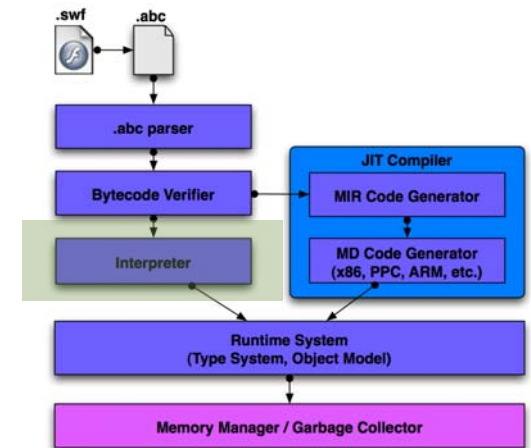
# AVM2 Interpreter

- Stack oriented
  - no surprises here; non-threaded
  - all values boxed; 32bit encoding
  - executes directly from verified buffer
  - no bytecode modification



# AVM2 Just-in-Time Compiler (JIT)

- First pass
  - intermediate representation (MIR) produced concurrent with verification
  - early binding
  - constant folding
  - copy and constant propagation
  - common sub-expression elimination (CSE)
- Second pass
  - native code generation
  - instruction selection
  - register allocation
  - dead code elimination (DCE)



# Tamarin

- AVM2 released to open source community

The goal of the "Tamarin" project is to implement a high-performance, open source implementation of the ECMAScript 4th edition (ES4) language specification. The Tamarin virtual machine will be used by Mozilla within SpiderMonkey, the core JavaScript engine embedded in Firefox®, and other products based on Mozilla technology. The code will continue to be used by Adobe as part of the ActionScript™ Virtual Machine within Adobe® Flash® Player.

- Working intimately with Mozilla to integrate with JavaScript engine
- See for yourself
  - <http://www.mozilla.org/projects/tamarin/>

# Interesting papers

- M. Chen, K. Olukotun, “Targeting Dynamic Compilation for Embedded Environments”
  - Computer System Lab, Stanford University
- M. Poletto, V. Sarkar, “Linear Scan Register Allocation”
- U. Hölzle, D. Ungar, “Optimizing Dynamically-Dispatched Calls with Run-Time Type Feedback”
  - Computer System Lab, Stanford University / Sun Microsystems
- J. Aycock, “A Brief History of Just-in-Time”

**Better by Adobe.™**