Weaving Code Extensions into JavaScript

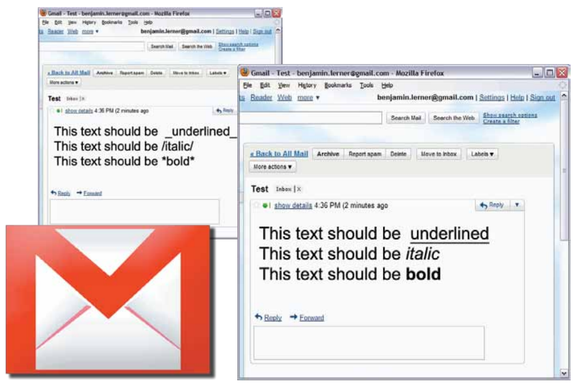
Benjamin Lerner, Herman Venter & Dan Grossman

University of Washington *MSR Redmond* *University of Washington*

Customizing the web: apps and browsers

Browsers and web apps get modified in unexpected ways by third parties: ***400 million*** Firefox installations use add-ons daily. Yet the extension ***mechanisms*** are ***brittle and semantically broken!***

"I get a lot of plain-text email, and wish Gmail would format it nicer"



"Firefox's new-tab screen is blank, so let's make it more useful"

Replacing idioms with aspects

```
eval("foo = " +
      foo.toString()
        .replace("some code",
                 "modified code"));
```

Create a new closure and bind to existing name

A closure's toString() returns its source code

String-level search & replace

```
function onLoad(evt)  { window.alert("hello"); }

window.addEventListener("load", onLoad, ...);

eval("onLoad = " +
      onLoad.toString.replace("hello",
                              "hi there"));

> ...loading the page...
> Alert: "hello" – wrong answer!
```

"Monkey-patching" breaks aliases, scoping, and possibly introduces syntax errors

```
at pointcut(callee(window.alert) &&
            within(onLoad))

before(msg) {
  msg = "hi there";
}
```

Pointcut: when to run the advice

Arguments to function are advice parameters

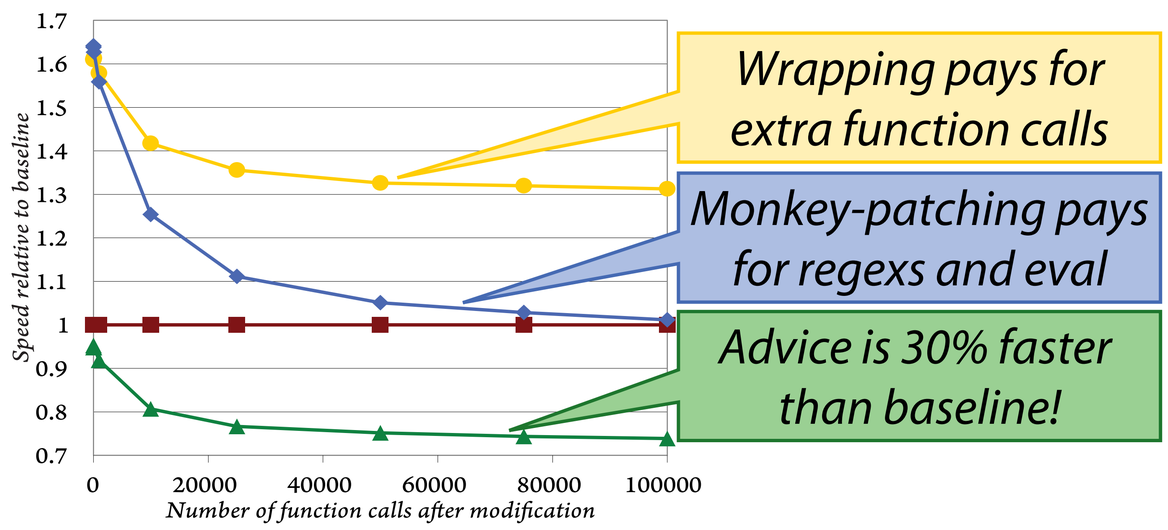
Advice: what new code to run at each pointcut

The same patch with aspects is shorter, cleaner, faster, and more correct

Implementation & Effectiveness

10% of top 350 Firefox extensions use monkey-patches. Examining 20 of those in detail:

- 99 KLOC total
- 2.7 KLOC patches
- We can easily express ***621/636 patches***
- Remaining 15 are either easily fixable ***or are bugs***



Dynamic advice weaving works particularly well with JIT compilation: just delete the cached function and re-JIT. And with very minimal runtime support, we get nearly ***30%*** performance gains over pure JS, and ***60—70%*** improvements over common idioms!