

SATToSE 2009

End-user Web Development Lively Fabrik and Lively Wiki

Jens Lincke

Robert Krahn

Robert Hirschfeld

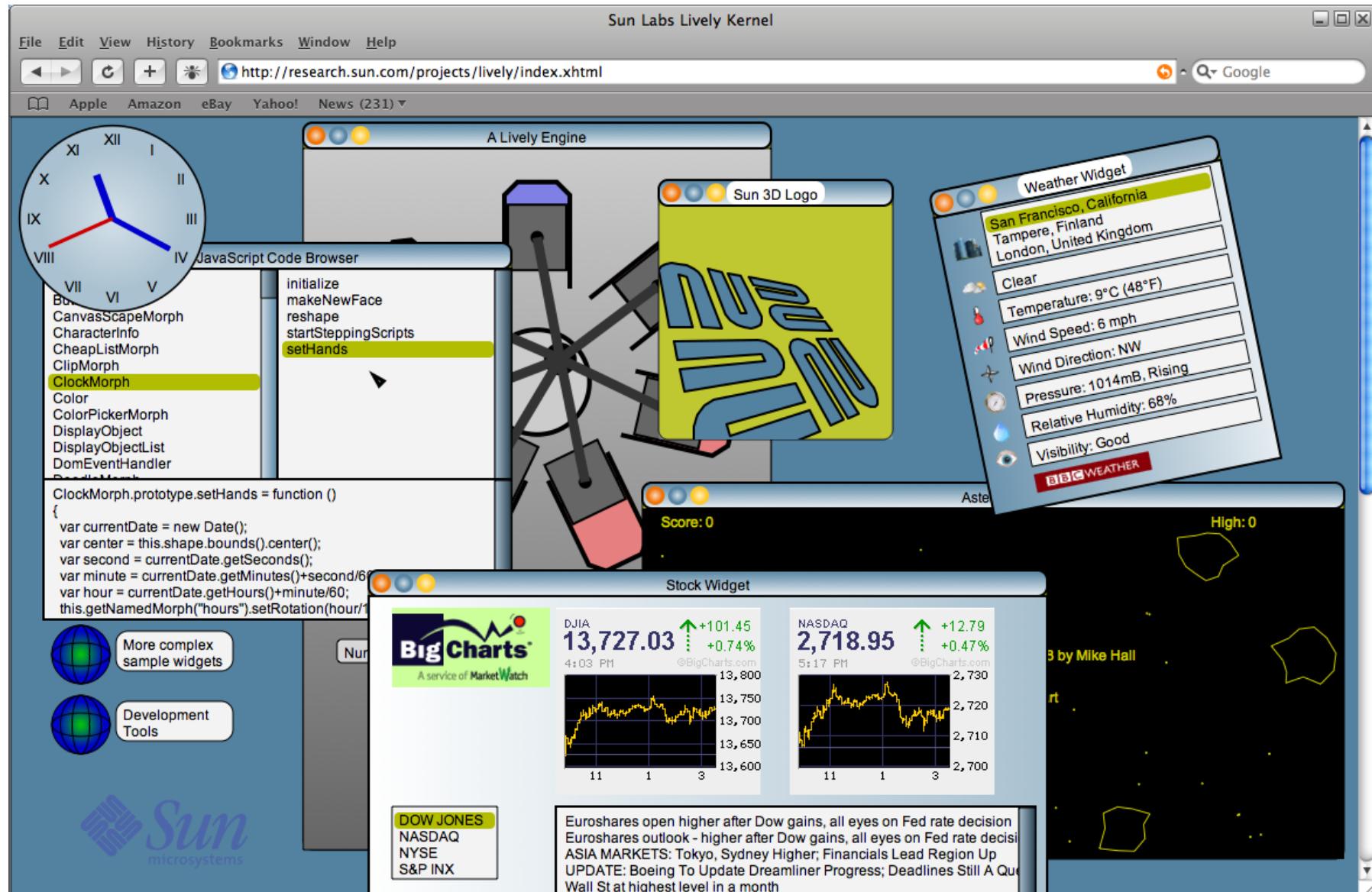
Hasso-Plattner-Institut Potsdam
Software Architecture Group

<http://www.hpi.uni-potsdam.de/swa/>

Motivation

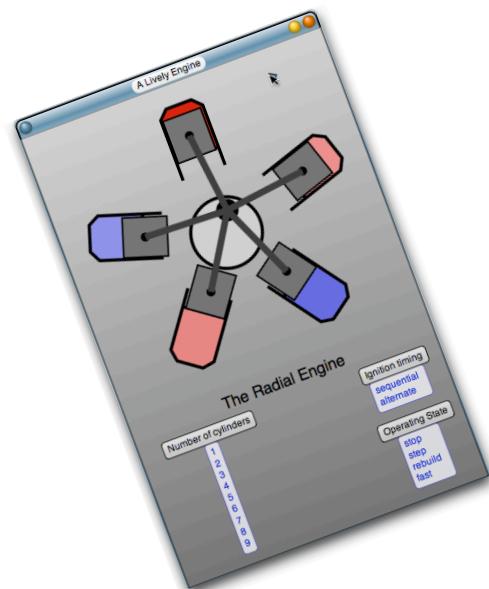
- End-users treat Web browser as OS
- EUD scenarios
 - Customization of existing Web applications
 - Widgets
 - Mashups
 - Simple Web Applications
- Growing Web programming capacities

Background: Lively Kernel



Lively Kernel Vision

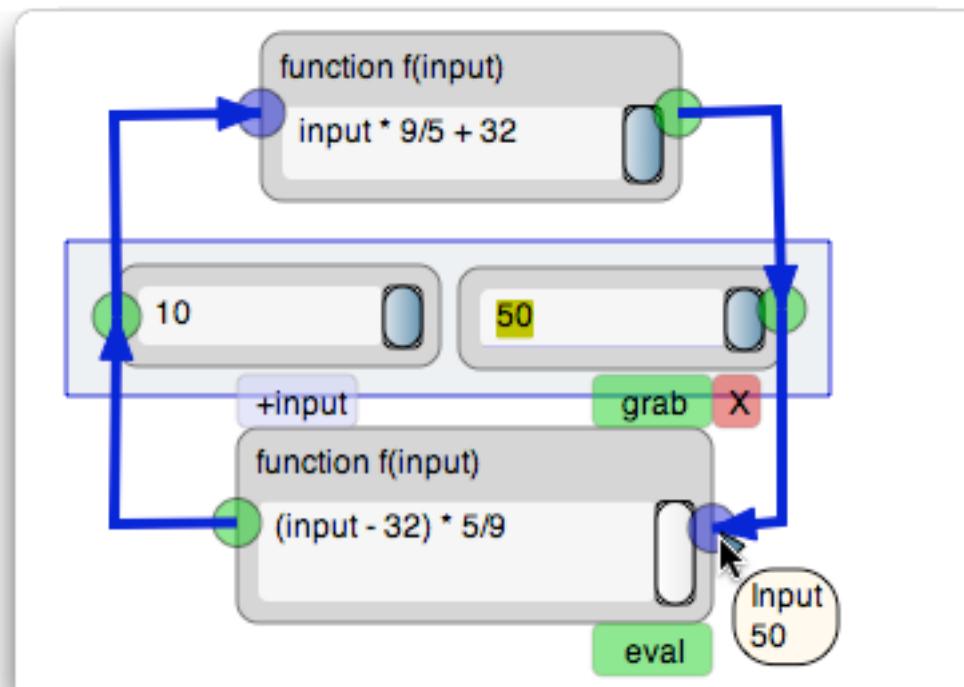
A wiki of Active Objects that can be
programmed in JavaScript
by wires and tiles



[Dan Ingalls]

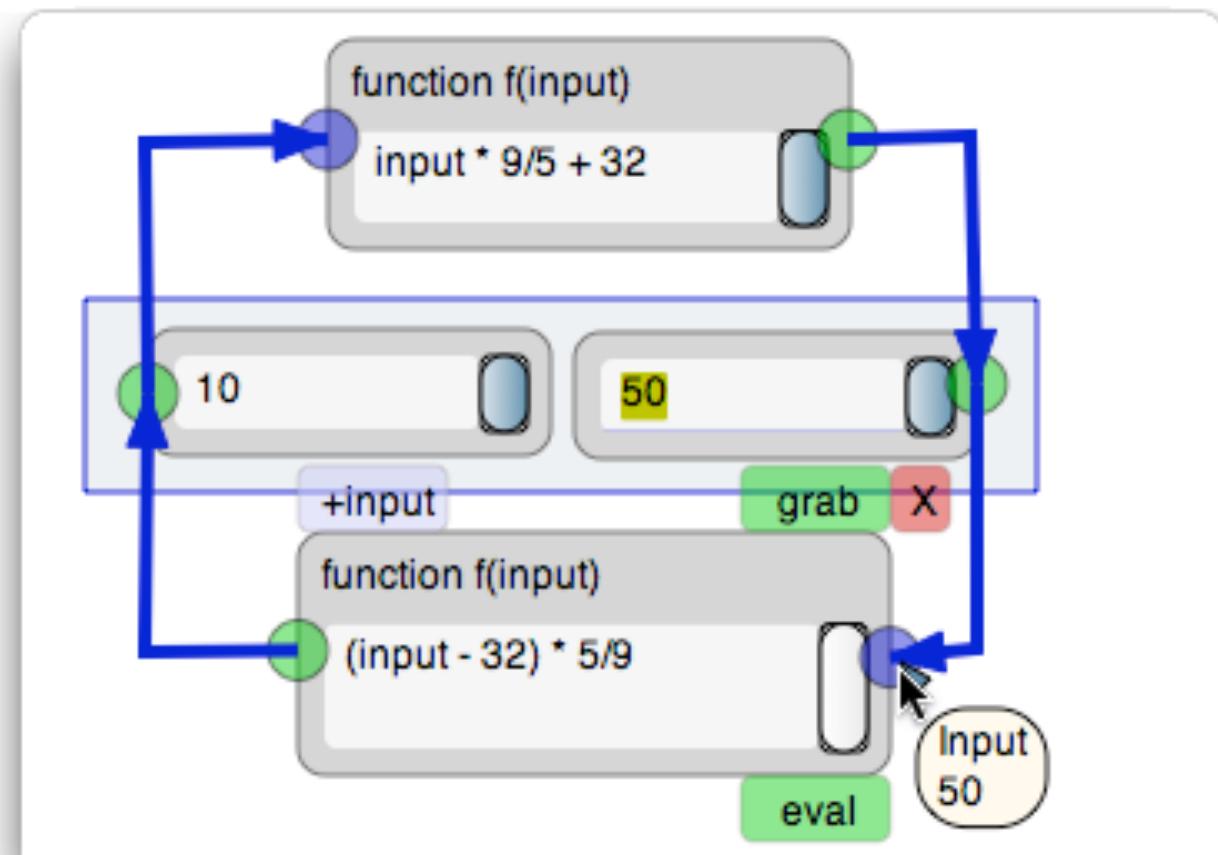
Lively Fabrik

- Empower end-users to create dynamic Web-applications with Mashups and Active Objects
- Combine Fabrik and Lively Kernel

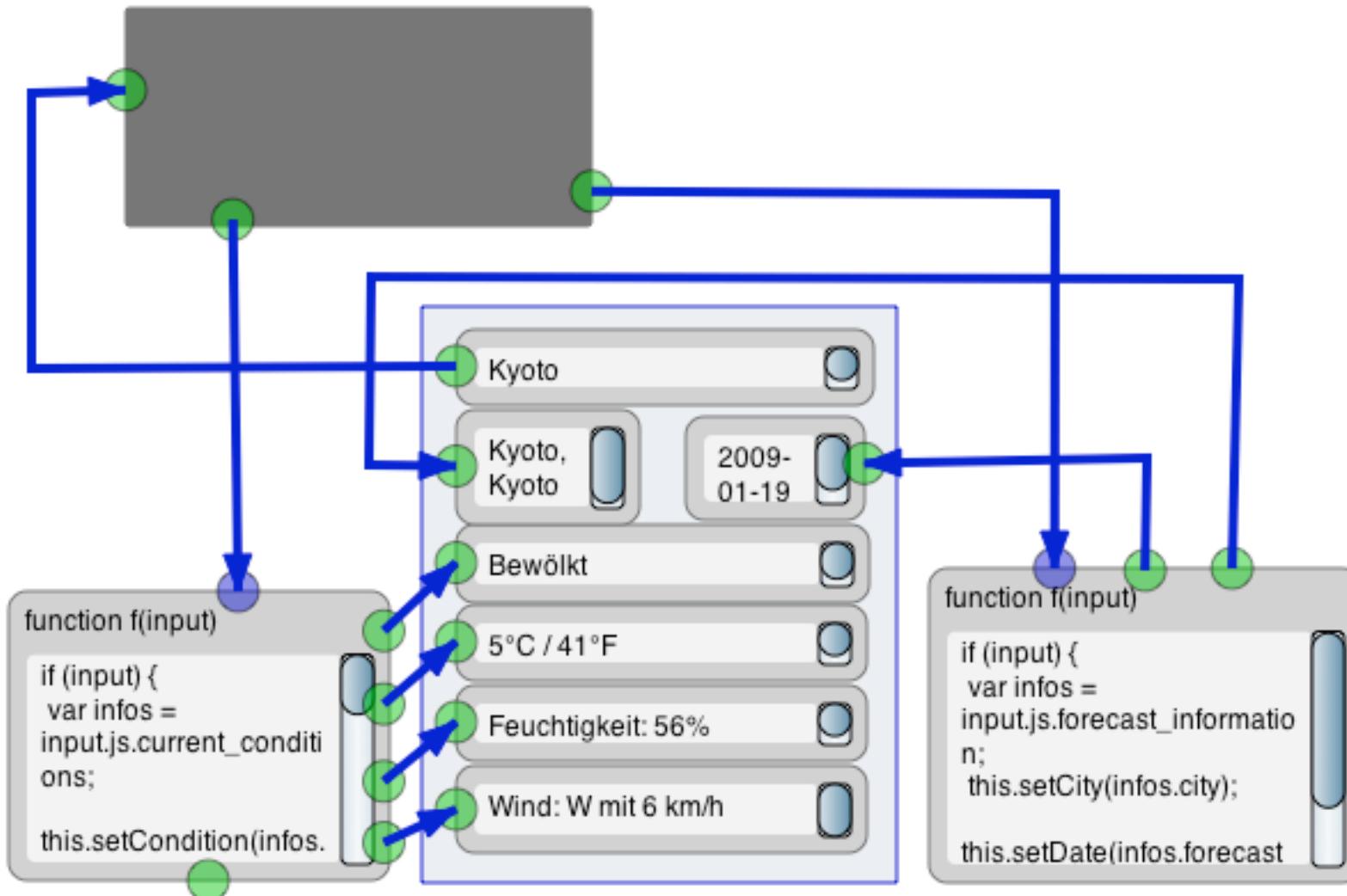


User Interface

- Combine UI and Behavior
- Continuous Running
- Visual Aids



Example Weather Widget



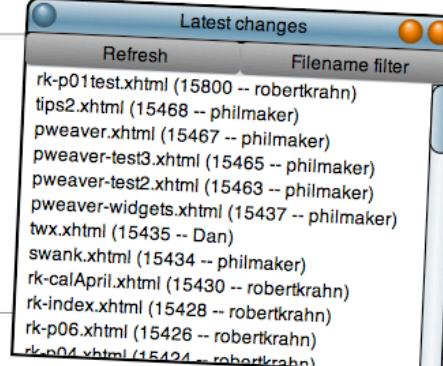
Lively Wiki A Development Environment for Creating and Sharing Rich Web Content

- Robert Krahn's Master's thesis
- Wiki principles: open, incremental, organic
- Developing *inside the Web for the Web*

Welcome to the Lively Kernel Wiki

Introduction

The Sun Labs Lively Kernel is a web programming environment developed at Sun Microsystems Laboratories. The Lively Kernel supports desktop-style applications with rich graphics and direct manipulation capabilities, but without the installation or upgrade hassles that conventional desktop applications have. The system is written entirely in the JavaScript programming language, a language supported by all the web browsers, with the intent that the system can run in commercial web browsers without installation or any plug-in components. The system leverages the dynamic characteristics of the JavaScript language to make it possible to create, modify and deploy applications on the fly, using tools built into the system itself. In addition to its application execution capabilities, the Lively Kernel can also function as an integrated development environment (IDE), making the whole system self-sufficient and able to improve and extend itself dynamically.



Latest changes

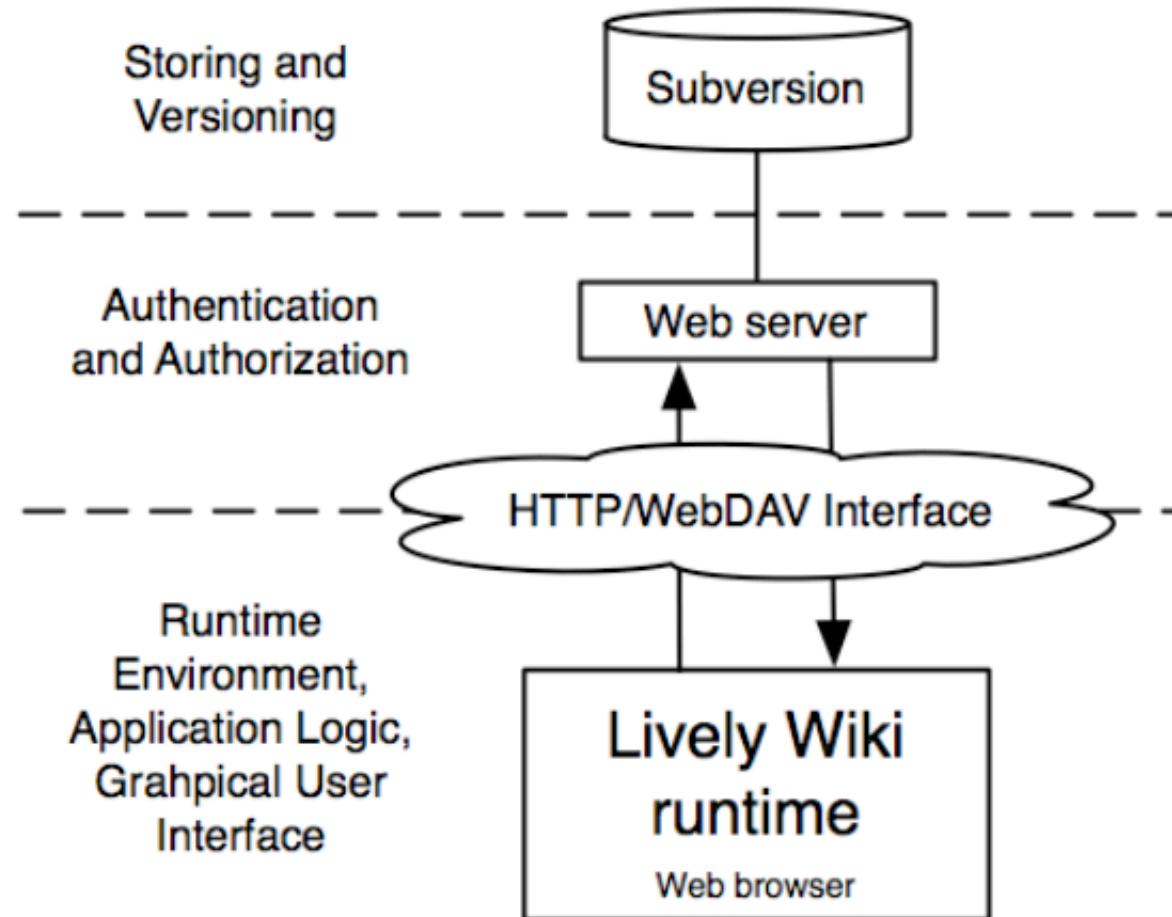
Refresh	Filename filter
rk-p01test.xhtml (15800 -- robertkrahn)	
tips2.xhtml (15468 -- philmaker)	
pweaver.xhtml (15467 -- philmaker)	
pweaver-test3.xhtml (15465 -- philmaker)	
pweaver-test2.xhtml (15463 -- philmaker)	
pweaver-widgets.xhtml (15437 -- philmaker)	
twx.xhtml (15435 -- Dan)	
swank.xhtml (15434 -- philmaker)	
rk-calApril.xhtml (15430 -- robertkrahn)	
rk-index.xhtml (15428 -- robertkrahn)	
rk-p06.xhtml (15426 -- robertkrahn)	
rk-p04.xhtml (15424 -- robertkrahn)	



Useful Links:

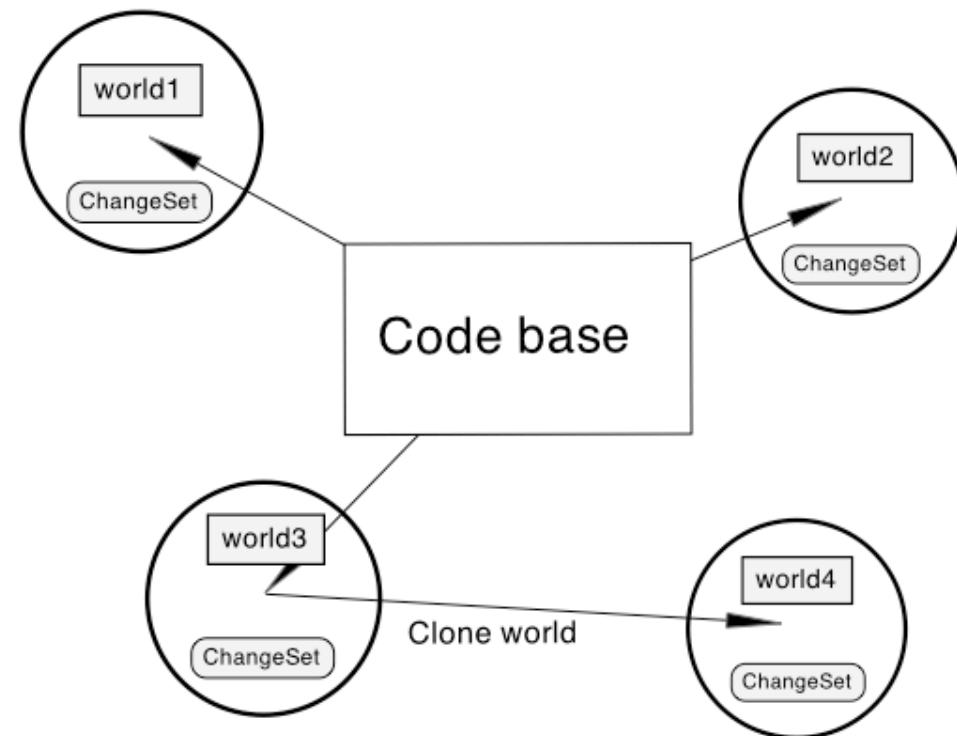
- [Dan's Kaleidoscope](#)
- [Tutorial: Programming in the wiki](#)
- [Lively Examples](#)
- [Wiki control scripts](#)
- [Fabrik playground](#)
- [Engine example](#)
- [Simple objects](#)
- [Project Page at Sun](#)

Lively Wiki Architecture



Web Programming Model

- Network of Lively Kernel worlds
 - Versions
 - Links
- Development tools
 - JavaScript and beyond



Development Tools

ide.js:lively.ide.SystemBrowser

Base.js (not loaded)
Contributions.js (not loaded)
Core.js
Data.js (not loaded)
defaultconfig.js (not loaded)
demofx.js (not loaded)
Examples.js (not loaded)
Fabrik.js (not loaded)
GridLayout.js (not loaded)
Helper.js (not loaded)
ide.js

classes functions objects

Load all LineNo Refresh

```
ide.BasicBrowser.subclass('lively.ide.SystemBrowser', {

documentation: 'Browser for source code parsed from js files',
viewTitle: "SystemBrowser",

initialize: function($super) {
  $super();
  this.installFilter(new lively.ide.NodeTypeFilter(lively.ide.Cl
},

rootNode: function() {
  ide.startSourceControl();
  if (!this._rootNode)
    this._rootNode = new ide.SourceControlNode(tools.Sourc
// this._rootNode = new ide.EnvironmentNode(Global, this

```

lively.ide.BasicBrowser
lively.ide.BrowserNode
lively.ide.BrowserCommand
lively.ide.NodeFilter
lively.ide.SortFilter
lively.ide.NodeTypeFilter
lively.ide.SystemBrowser
lively.ide.SystemBrowser
lively.ide.SourceControlN
lively.ide.FileFragmentNo
lively.ide.CompleteFileFr
lively.ide.CompleteOmeta

documentation (proto)
viewTitle (proto)
initialize (proto)
rootNode (proto)

TestRunner

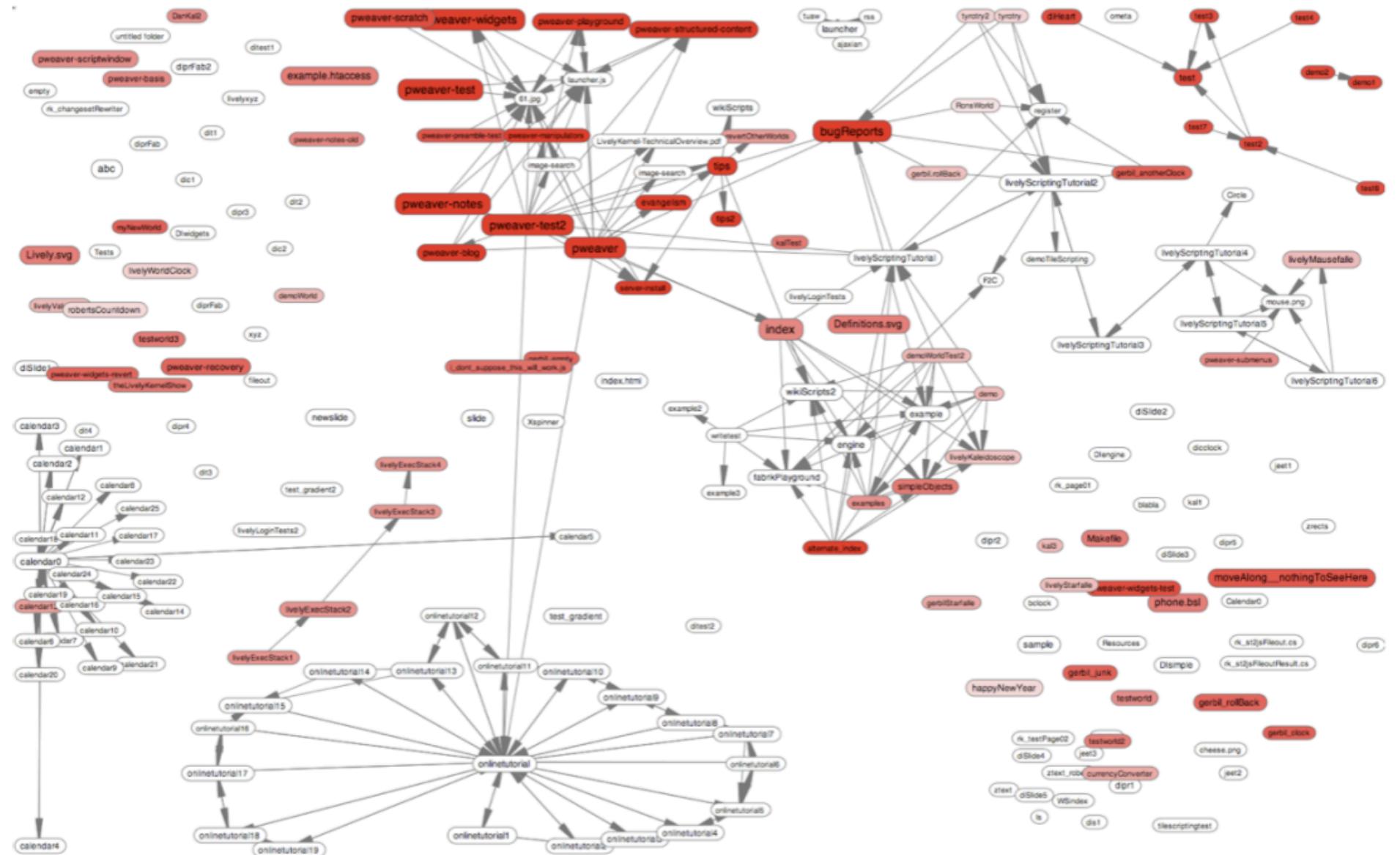
AComponentCopierTest (1547ms)
AFabrikSerializationTest (2476ms)
AFabrikUITest (75ms)
AScrollPaneTest (29ms)
ASerializationTest (192ms)
Alively.Tests.CoreTest.CopierTest (21ms)
ButtonMorphTest
ClassTest (175ms)
ComponentModelTest (16ms)
ComponentMorphTest (363ms)
ComponentSerializeTest
ComponentTest (150ms)
ConnectorMorphTest (60ms)
DomRecordTest (1ms)
FabrikComponentConnectionTest (53ms)
FabrikComponentTest (4ms)
FabrikComponentTest (4ms)

Run Tests Run All Tests

Tests run: 413 -- Tests failed: 15 -- Time: 33.414s

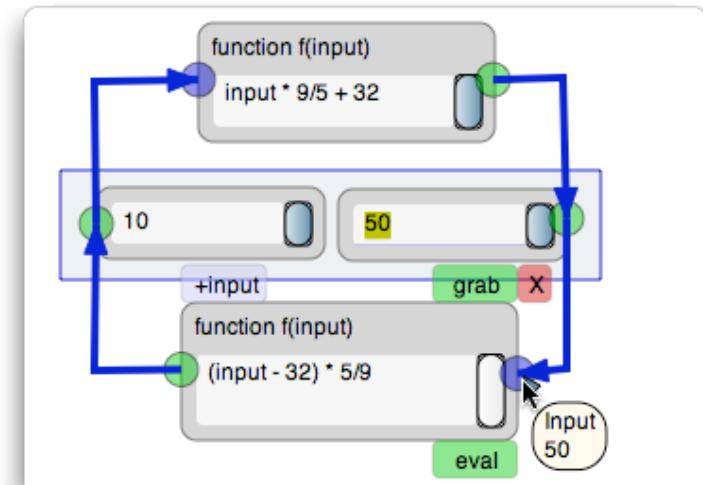
- >morph overlaps! (pt(180.0,100.0) != pt(181.0,101.0)) in TestFramework.js (Line 87)
- ComponentMorphTest.testAddButtonAndText
- >morph overlaps! (pt(180.0,100.0) != pt(181.0,101.0)) in TestFramework.js (Line 87)
- LayoutTests.testHLayoutCenterMorphs
- > assert failed (Morph: 6386:Morph([0,0,20,20]) overlaps its ownerBounds!) in TestFrame
- SyntaxHighlighterTest.testGetGrammarString
- >Maximum call stack size exceeded. in ometa-base.js (Line 114)
- SyntaxHighlighterTest.testCompileGrammar
- >Maximum call stack size exceeded. in ometa-base.js (Line 114)

Example: Network Visualization in Lively Wiki



Summary

- Web based end-user programming environment
- Lively Fabrik
 - Combine visual data flow with scripting
 - Create UI and behavior in one view
- Lively Wiki
 - Wiki principles: open, incremental, organic
 - Developing *inside the Web for the Web!*



SATToSE 2009

End-user Web Development Lively Fabrik and Lively Wiki

Hasso-Plattner-Institut Potsdam
Software Architecture Group

Prof. Dr. Robert Hirschfeld

Jens Lincke

Robert Krahn

<http://www.hpi.uni-potsdam.de/swa/>