Web-based Development Environments
Organization and Topics

Hasso-Plattner-Institut Potsdam

Software Architecture Group

Jens Lincke, Stefan Lehmann

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

2016

DEMO

New Architecture Decisions in Lively4

- Less forced structure
- As single page web application

- Use Branching to work in parallel
- Embrace HTML as Morphic-like UI

- Tweak every website with Lively4
- Access resources through unified file API

Scenarios

- Lively:
  - Creation of active content on a web page
  - through object composition, scripting, and direct manipulation
  - Example: Lively 1-3
- Web Service Hub:
  - Program the synchronization of websites and cloud storages
  - Example: Github, Wunderlist, and Trello issues in sync

Scenarios

- Cross-site Annotations:
  - Text Highlighting and annotations across websites
  - Fuzzy matching based on content
  - Shared with other users
  - Persisted on cloud storages
  - Example: Blendle, blogspot, e-books
- Vacation Map:
  - Use semantic annotations on websites
  - Enrich website with own UI
  - Synchronize extracted content with web storages
  - Mesh up information heterogeneous information on a map
  - Example: tripadvisor, wikidata, eventim

Lively4
Technologies

Development Approach

- Live
- S3
- Web-based
  - On any Website
- Collaborative
  - sync
  - async
- Testing
  - CI
  - live

Language Support

- ES6 Modules
- Source Code-based Workflow
  - .html, .md, .js
  - Multi-language System
    - Runtime Adaptation
    - Debugging
    - Projectional Editing
    - Web Assembly Modules

Language Extensions

- Active Expressions
- COP
- OCP

Browser Support

- Semantic Web
- Windows App

Runtime Technologies

- Unified Data Access
- Cloud Variables
- WebStrates

Cloud Programming

- Project Deployment
- Cloud Dataflow
- Service Deployment

Performance

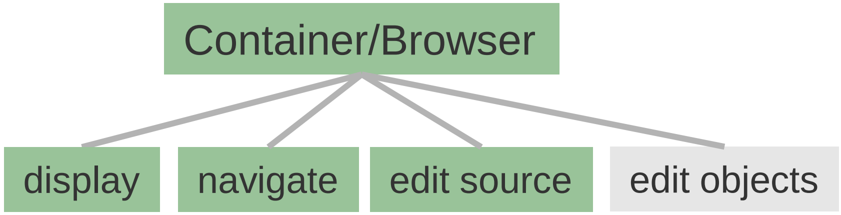
- Offline First
- Cache & Sync
- Cached Transformation

UI Technology

- HTML as Morphic
- Web Components
- Monkey Patching
  - by example
  - for testing
- Data Bindings

Lively4
Components

Tools



File System Control

Sync Tool

Component Bin

(Group) Object Editor

Workspace

Test Runner

Debugger

X-ray

Meta Tools

Context Menu

Halo

Object Lasso

Widgets

HTML Elements

ACE Editor

d3

Graph Layout

TOPICS

Topics

- **Cache**: Change-aware File Caching
- **CVar**: Cloud Variables
- **AExpr**: Active Expressions
- **COP**: COP 2.0
- **Changes**: Change-based Persistence
- **Indexing**: Personal Cloud Search
- **Services**: Access Web Services
- **MPatch**: Lively Changes of Foreign Sites
- **RDF**: Semantic Web Scripting
- **Wasm**: Web Assembly
- **ProjE**: Projectional Editor
- **Edge**: MS Edge + Windows APP

Project: **Cache**

Change-aware File Caching

- Background:
  - Inherent conflict between developer mode and fast page loading
- Problem:
  - Browser caches: Current version is outdated
  - Server-side caches: github.io cache 1 – 10 minutes
  - Time-consuming compilation process
- Ideas:
  - Caches and Hashes
    - Cloud Storages and Services can provide you with metadata about recent changes
    - Meta information can invalidate caches, e.g. dropbox hashes, github version numbers
  - Cache compile results and update when source caches (browser/cloud watcher)
- Goal:
  - Combine benefits of caching with up-to-date content
    - Always get latest version of file when changed
    - Only transport file when it actually changes
  - Minimal amount of recompilation

Project: **CVar**

Cloud Variables

- Background:
  - Programming distributed or multi-user applications, like games, involve complex logic
- Idea:
  - Provide developers with simple, replicated storage abstraction that can be used like ordinary variables
- Related work:
  - Liquid Storage
  - cloud data in Touch Develop
- Goals:
  - Provide simple-to-use cloud variable API
  - Prototype sample application

Project: **AExpr**

Active Expressions

- Background: Using imperative JavaScript to
  - Derive declarative constraint descriptions (OCP)
  - Derive groups of objects
- Example:

```
select(Morph, m => m.color === ‘green’)
```

- Goal:
  - Extract active expressions into reusable library
  - Extend library to track DOM changes
  - Example Application: Visual debug information for object groups

Project: **COP**

ContextJS 2.0

- Background: ContextJS
  - Context-oriented Programming for JavaScript
  - Developed and used as part of Lively Kernel
- Idea:
  - From context-dependent method invocations to objects are in a context
- Goal:
  - Make context activation explicit
    - To allow arbitrary partial behavior like triggers, connections, constraint, events
    - Various activation mechanisms
  - Optimize for performance

Project: **Changes**

Change-based Persistence

- Background:
  - Asynchronous collaboration requires fast update cycles
  - Transmitting full documents is easy but relatively slow
- Idea:
  - Transmitting and persisting changes will provide base technology for fine-granular undo and synchronous collaboration
- Related work:
  - Operational Transformation in Webstrates
    https://github.com/cklokmose/Webstrates
- Goal:
  - Synchronous collaborative editing of content on a lively page
  - Recording of fine-granular content edits provides undo

Project: **Indexing**

Personal Cloud Search

- Background:
  - Hard to search find private content and code
- Problem:
  - Cannot use public search services for private data
- Idea:
  - Search private data on cloud storages and services
  - Persist index on private services, cloud storage, or web browser
- Related work:
  - http://lunrjs.com/
  - https://github.com/fergiemcdowall/search-index
- Goal:
  - Provide instant code search in lively tools
  - Provide navigation between #tags in private content

Project: **Services**

Access Web Services

- Background:
  - Many easy to deploy but not live programmable web services, e.g. Travis
- Related Work:
  - Live Programming of node.js servers on lively-web.org
- Idea:
  - Easy Web Service Deployment
- Goal:
  - Provide infrastructure to deploy and update to remote services
  - Tool support (Web Service Workspace)

- Future work: IFTTT with lively4

Project: **MPatch**

Lively Changes of Foreign Sites

- Background:
  - Lively4 loadable on every website through Chrome extension
- Problem:
  - Content on web pages changes
  - Not every object on a web page has a unique identity
  - Interactive modification of DOM through Halo is not persistent
- Idea:
  - Capturing Edits
  - Identify object through patterns (Lively4 by Example)
- Related Work:
  - GreaseMonkey
- Goal:
  - Capture deletes/rearranges/styling/addition of content with Halo
  - Replay changes on future visits

Project: **RDF**

Semantic Web Scripting

- Background:
  - Websites provide semantic information about displayed content
- Idea:
  - Reify semantic information and make them progammable
- Example: Personal interactive ScrapMap for vacation using tripadvisor, wikipedia, eventim
- Goal:
  - Extract RDFa Data, Microformats
  - Embed UI for Publish/Share
- Related Work:
  - https://developers.google.com/structured-data/testing-tool/

Project: **Wasm**

Web Assembly

- Background: Web Assembly
  - Low-level language to run in browsers
  - Compile target
  - Easier to parse and fast to execute
- Problem:
  - How to integrate Web Assembly-based development in live environment
- Goal:
  - Editing and execution of <Bouncing Ball> written in C/Ruby/… on a web page
  - Using a C/Ruby/… as module from Lively4

Project: **ProjE**
Projectional Editor

- Background:
  - Projectional editing help users to make less syntax error and enables domain-specific views on source code
- Problem:
  - JavaScript’s dynamic nature makes static inference, auto completion, syntax completion hard
- Idea:
  - Use static inference to provide enough static information for projectional editing
- Goal:
  - Projectional editor for creating (d3 visualizations | …)
- Related Work:
  - Tile Script
  - Touch Develop
  - GP
  - http://sevin7676.github.io/Ace.Tern/demo.html

Project: **Edge**

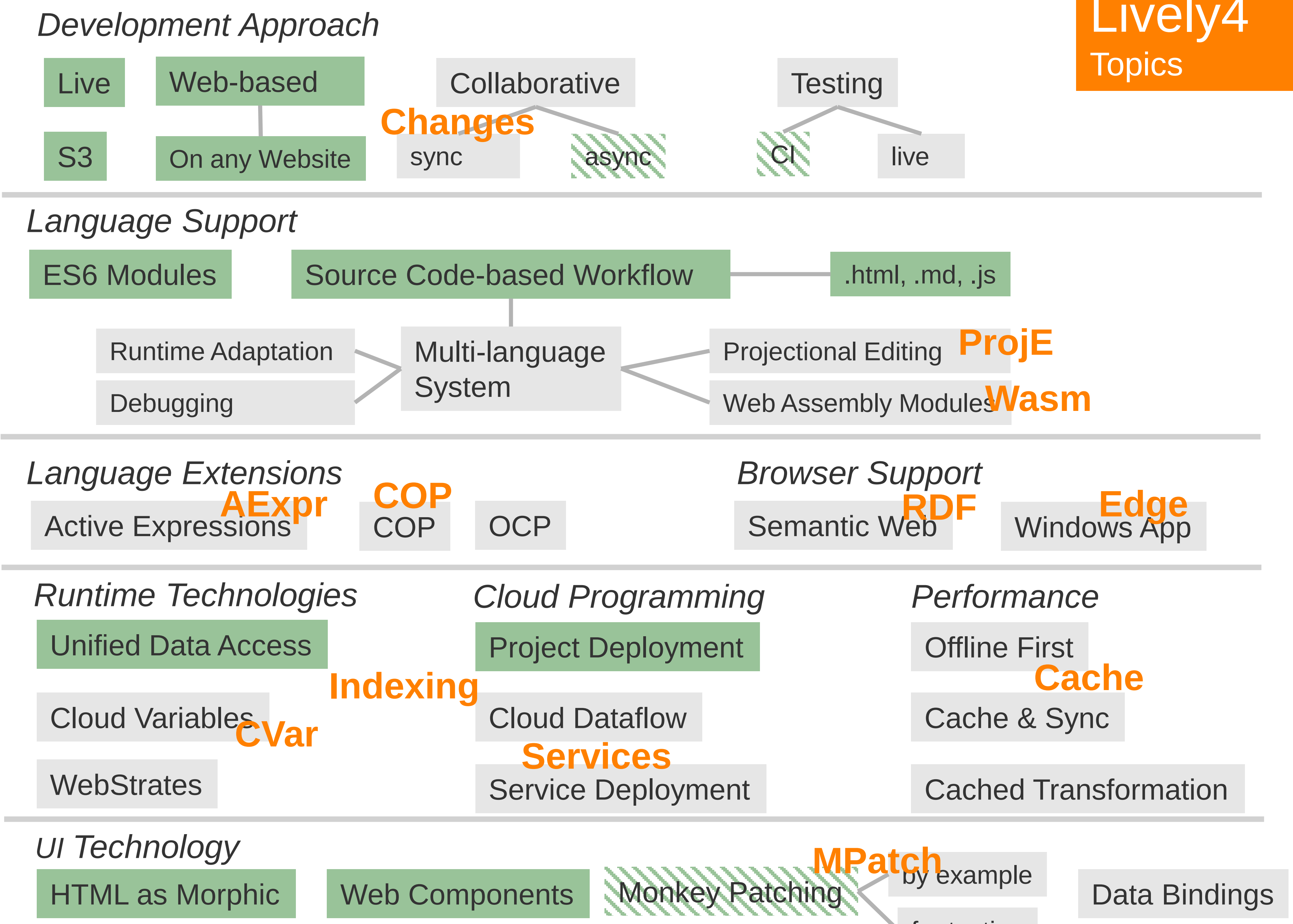
Lively4 on MS Edge + Windows APP

- Background:
  - Windows JavaScript apps provide most sophisticated API for hand-writing recognition
- Problem:
  - New web technologies are not adapted equally in all browsers, e.g. missing Service Worker in MS Edge
- Idea:
  - Use polyfills to run Lively4 on MS Edge
- Goal:
  - Run Lively4 in a Windows JavaScript app
  - Run Tim’s hand-writing recognition Demo in Lively4

Topics

- **Cache**: Change-aware File Caching
- **CVar**: Cloud Variables
- **AExpr**: Active Expressions
- **COP**: COP 2.0
- **Changes**: Change-based Persistence
- **Indexing**: Personal Cloud Search
- **Services**: Access Web Services
- **MPatch**: Lively Changes of Foreign Sites
- **RDF**: Semantic Web Scripting
- **Wasm**: Web Assembly
- **ProjE**: Projectional Editor
- **Edge**: MS Edge + Windows APP

Lively4
Topics



Development Approach

- Live
- S3
- Web-based
- On any Website
- Collaborative
- Changes
- sync
- async
- Testing
- CI
- live

Language Support

- ES6 Modules
- Source Code-based Workflow
- .html, .md, .js
- Multi-language System
- Runtime Adaptation
- Debugging
- Projectional Editing
- ProjE
- Web Assembly Modules
- Wasm

Language Extensions

- Active Expressions
- AExpr
- COP
- COP
- OCP

Browser Support

- Semantic Web
- RDF
- Windows App
- Edge

Runtime Technologies

- Unified Data Access
- Cloud Variables
- CVar
- WebStrates
- Indexing

Cloud Programming

- Project Deployment
- Cloud Dataflow
- Service Deployment
- Services

Performance

- Offline First
- Cache
- Cache & Sync
- Cached Transformation

UI Technology

- HTML as Morphic
- Web Components
- Monkey Patching
- MPatch
- by example
- for testing
- Data Bindings

Development Links

- Github Projects:
  - https://github.com/livelykernel/lively4-core
  - https://github.com/LivelyKernel/lively4-server (optional)
  - https://github.com/LivelyKernel/lively4-chrome-loader (optional)
- Github Wiki
  - https://github.com/LivelyKernel/Lively4/wiki/WebDev16
- Deployed on github.io:
  - https://livelykernel.github.io/lively4-core/draft/start.html?load=https://lively4/README.md
- Deployed on lively-kernel.org:
  - https://lively-kernel.org/lively4/lively4-core/draft/start.html

Getting Started

Basic Lively4 development workflow:

a) Go to https://lively-kernel.org/lively4/lively4-core/draft/start.html

b) Play around with objects and in workspace

   – Ctrl-Left/Right Click for Halo and context menu

c) Use tools to browse and change modules and templates

d) Open Sync tool and log with github account

e) Press sync to update your instance and commit changes

Getting Started

Advanced Lively4:

- Work in different branches under https://lively-kernel.org/lively4
- Try out serverless variant https://livelykernel.github.io/lively4-core/draft/start.html?load=https://lively4//
- Install your own lively4-server

DEMO 2

DELIVERABLES/PROJECT

Organization

- Course
  - Weekly meetings (Slot to be found)
  - Project-Seminar, 4 SWS, 2 students per group
- Grading
  - 6 ECTS graded credit points
  - Grade based on project work and presentation
- Hand-In
  - Presentation, Screencast, Sourcecode
- Important dates
  - Project topics on April 13th
  - Enrollment with preferred topic names on or before April 19th
    - Mail to stefan.lehmann@hpi.de and jens.lincke@hpi.de with WebDev16 in subject
  - Topic assignment on April 20th
  - Presentation dates determined after topics are assigned

DEMO 3

Web-based Development Environments
Organization and Topics

Hasso-Plattner-Institut Potsdam

Software Architecture Group

Jens Lincke, Stefan Lehmann

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

2016