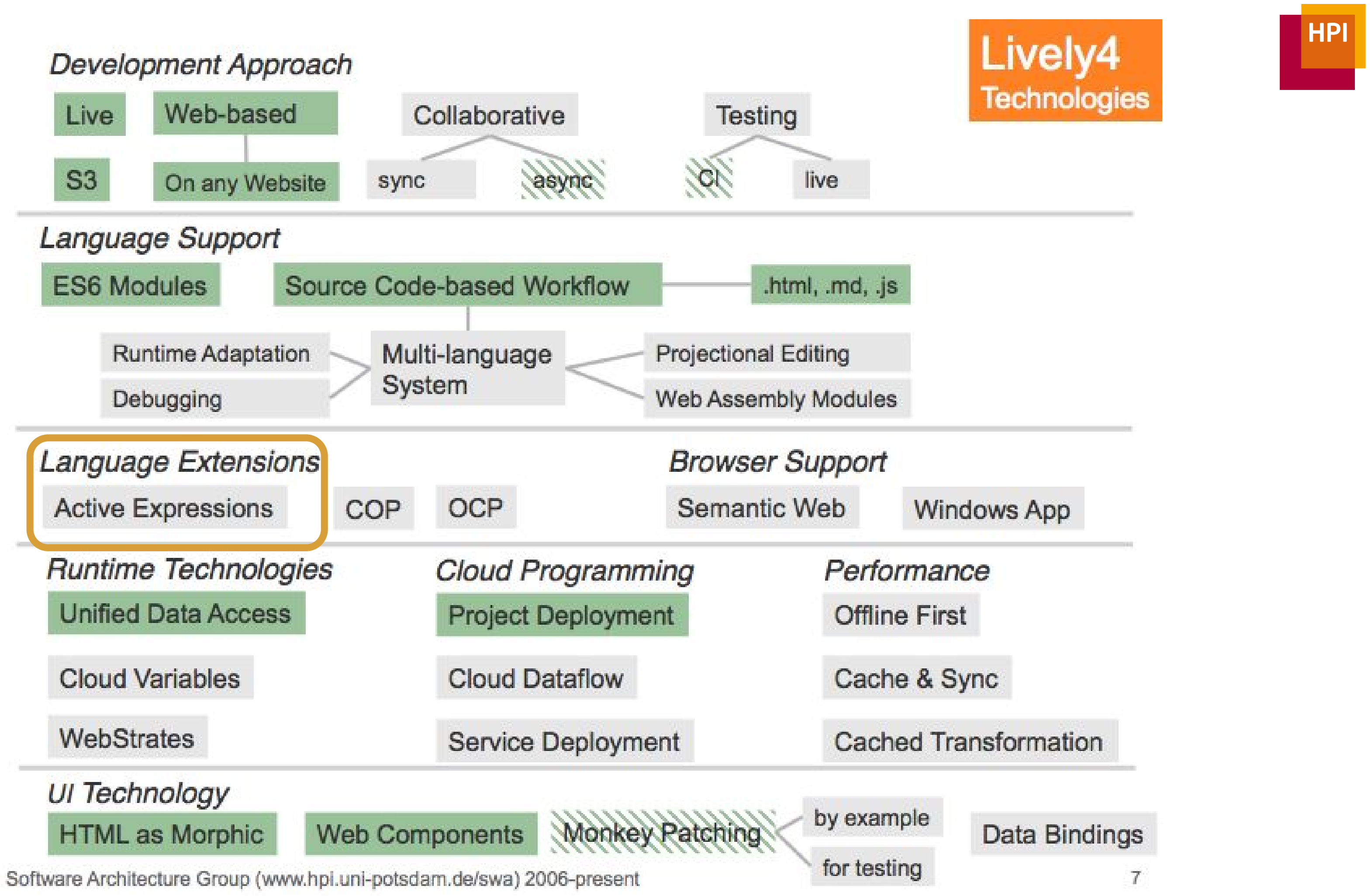
# Active Expressions

Web Development SS2016
Timo Djürken, Philipp Pajak



HPI

Lively4 Technologies

*Development Approach*

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

*Language Support*

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

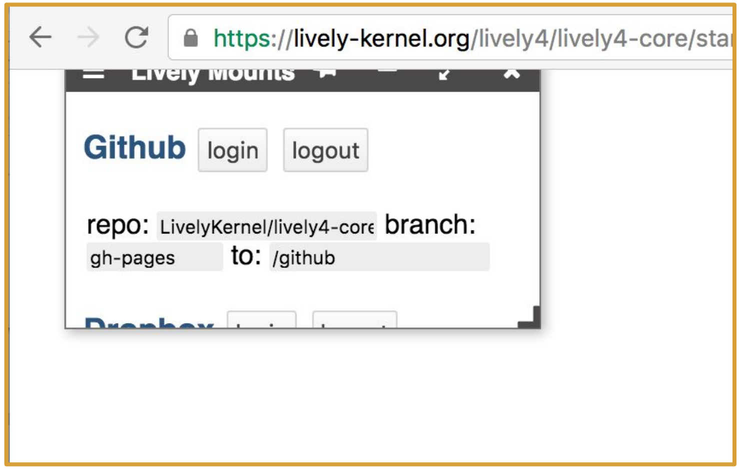
*UI Technology*

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

Software Architecture Group (www.hpi.uni-potsdam.de/swa) 2006-present

7

# Motivation


https://lively-kernel.org/lively4/lively4-core/sta

Lively Mounts

Github login logout

repo: LivelyKernel/lively4-core branch: gh-pages to: /github

**Problem:**

Window handles can **disappear** when moving **too far up**

## Motivation

### Naive “solution”:

```
setInterval(() => {
  document.querySelectorAll('lively-window').forEach(w => {
    requestAnimationFrame(() => {
      if (parseInt(w.style.top) < 0)
        w.style.top = 0;
    });
  });
}, 100);
```

# Motivation

## Active Expression:

```
new AExpr( win => parseInt(win.style.top)  < 0 )

  .applyOnAll( new ActiveDOMView('lively-window') )

  .onChange( win => win.style.top = 0 );
```

## Motivation

Using imperative JavaScript code to ...

- find **groups** of objects,
- **efficiently react** to changes,
- **separate** concerns,
- while keeping the code **readable** and **concise**

# ActiveExpressions are not …

… **Constraint Solvers**

but can supplement them

… **COP Layers**

but can be combined with them

… **Data Bindings**

but could implement them

# Constraints: Babelsberg.js

Given a constraint: a=2*b+c

Babelsberg solves constraint in **any direction**

ActiveExpressions can only solve in **one direction**

**but:** can be **combined**, e.g. to trigger constraint solving

## COP: Behaviour Adaptation

COP can **alter** and **extend behavior**

ActiveExpressions **react** to changes

**but:** COP and ActiveExpressions can be **combined**

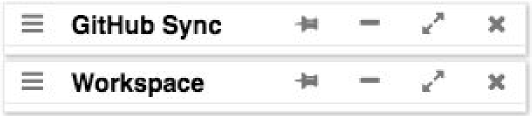
## Data Binding: AngularJS

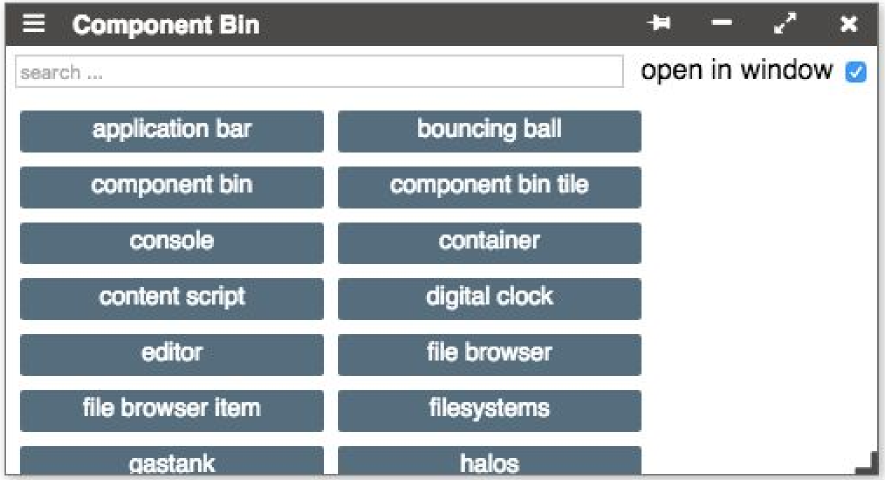
```
<input ng-model="firstname">
```

AngularJS **binds** models in **both directions**

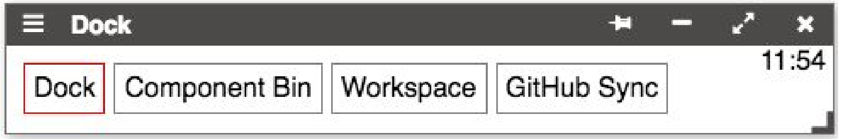
ActiveExpressions **could** implement similar bindings

**but:** two-way binding is **not main purpose** of ActiveExpressions





```
onTitleChange() {
   new AExpr(win => win.getAttribute('title'))
   .applyOnAll(this.activeWindowView)
   .onChange(win => {
     var windowTab = this.windowTabs.get(win);
     windowTab.innerHTML = win.getAttribute('title');
   });
 }
```



# Demo

## Implementation

**AExpr**

**Observes** given **conditions** and **triggers** on **changes**

**ActiveView**

**Collects** elements and **updates** itself when new elements come in

# AExpr: Condition Parsing

```
let outOfScreen = new AExpr(
    window => {
       return parseInt(window.style.top) < 0 || parseInt(window.style.left) < 0;
    }
 );
```

## AExpr: Condition Parsing

```
let outOfScreen = new AExpr(
   window => {
     return parseInt(window.style.top) < 0 || parseInt(window.style.left) < 0;
   }
);
```

1. Parsing the **AST** with acorn.js

# AExpr: Condition Parsing

```
let outOfScreen = new AExpr(
  window => {
    return parseInt(window.style.top) < 0 || parseInt(window.style.left) < 0;
  }
);
```

1. Parsing the **AST** with acorn.js
2. Match **context** variables

# AExpr: Condition Parsing

```
let outOfScreen = new AExpr(
   window => {
     return parseInt(window.style.top) < 0 || parseInt(window.style.left) < 0;
   }
 );
```

1. Parsing the **AST** with acorn.js
2. Match **context** variables
3. Collect relevant **properties**

## AExpr: Applying to Objects

```
/* single objects */
expr.applyOn( jsObjectA );
expr.applyOn( document.querySelector('#container') );


/* collections */
expr.applyOnAll( [jsObjectA, jsObjectB] );
expr.applyOnAll( new ActiveDOMView('div.ball') );
expr.applyOnAll( document.querySelectorAll('div.ball') );
```

# AExpr: Watching for Changes

- for JS **properties**:
  - **override** getter/setter

```
var newSetter = function(newValue) {
   if (this.__lively_expr_setters[variable]) {
     this.__lively_expr_setters[variable]
            .call(this, newValue);
   } else {
     this.__lively_expr_vars[variable] = newValue;
   }

   // alert watchers
   this.__lively_expr_watchers.forEach((w) => {
     w.test();
   });
 };
```

- for HTML **attributes**:
  - **MutationObserver** with filter

## ActiveDOMView: Implementation

```
class ActiveDOMView extends ActiveView {
  constructor (selector) { /* ... */ }
  onEnter (callback) { /* ... */ }
  onExit  (callback) { /* ... */ }
}
```

- **ActiveDOMView** is a concrete implementation of ActiveView
- Uses a **MutationObserver** to track added and removed elements

## ActiveObjectView: Idea

```
new ActiveObjectView(SomeClass)
```

- live view of **class instances**
- could use **COP layers** to extend constructors
- Already implemented as Reactive Object Queries *

*https://github.com/onsetsu/active-collection-prototype

## Challenges

- **AST** interpretation
- Handling both **DOM** and **JS** objects
- No proper **object observer**
- AExpr and **object lifecycle**

## Future Work: Recursive Parsing

```
new AExpr(
   rect => rect.getArea() > 500
);
```

- recursively analyze called functions
- collect “hidden” properties to observe

# Future Work: Context Derivation

```
let foo = new Foo();

watch (foo.bar > 500) {
  foo.tooHigh = true;
}
```

Transpile

```
let foo = new Foo();

new AExpr(_var1 => _var1.bar > 500)
  .applyOn(foo)
  .onChange(function(_var1) {
     _var1.tooHigh = true;
  });
```

- introduce new **keyword**
- **transpile** and derive correct **context** variables

```
onTitleChange() {
   new AExpr(win => win.getAttribute('title'))
     .applyOnAll(this.activeWindowView)
     .onChange(win => {
       var windowTab = this.windowTabs.get(win);
       windowTab.innerHTML = win.getAttribute('title');
   });
 }
```

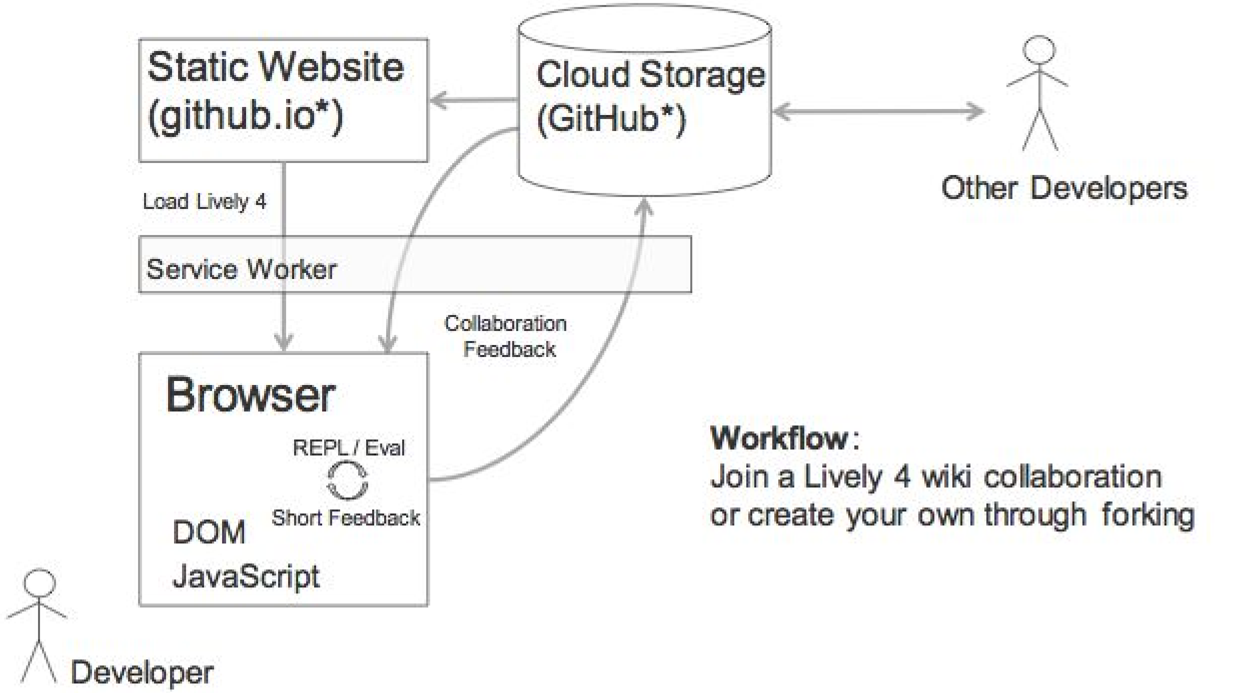
Recurive Parsing

Context Derivation

# Active Expressions

# Backups

# Lively 4

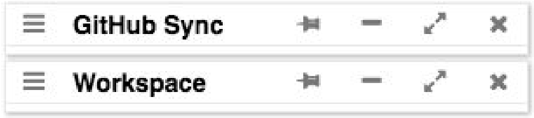


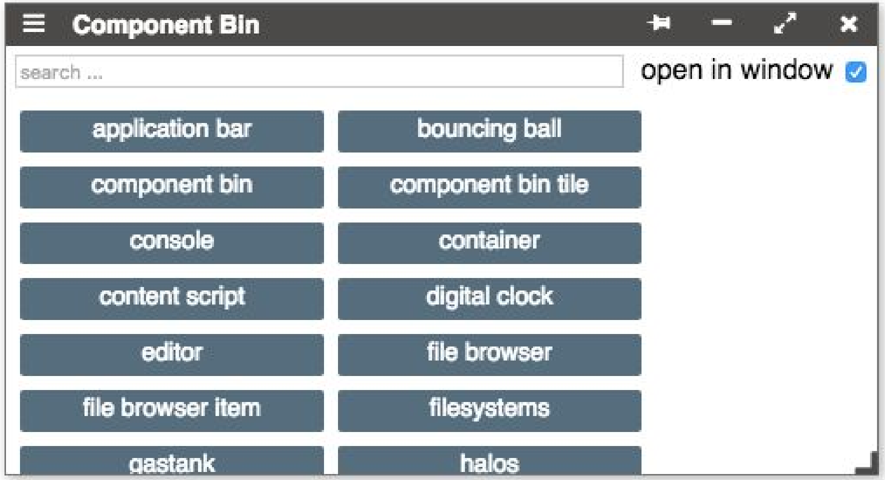
**Workflow**:
Join a Lively 4 wiki collaboration
or create your own through forking

* and others...

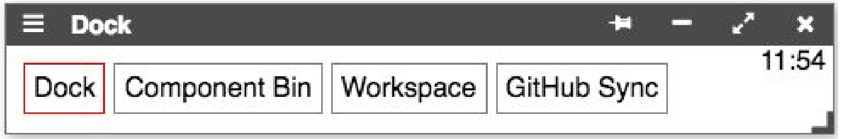
## Demo Ideen

- Inspector Frame
- Show bounding boxes
- **Window Dock**





```
onTitleChange() {
   new AExpr(win => win.getAttribute('title'))
     .applyOnAll(this.activeWindowView)
     .onChange(win => {
       var windowTab = this.windowTabs.get(win);
       windowTab.innerHTML = win.getAttribute('title');
   });
 }
```



## Active Expressions