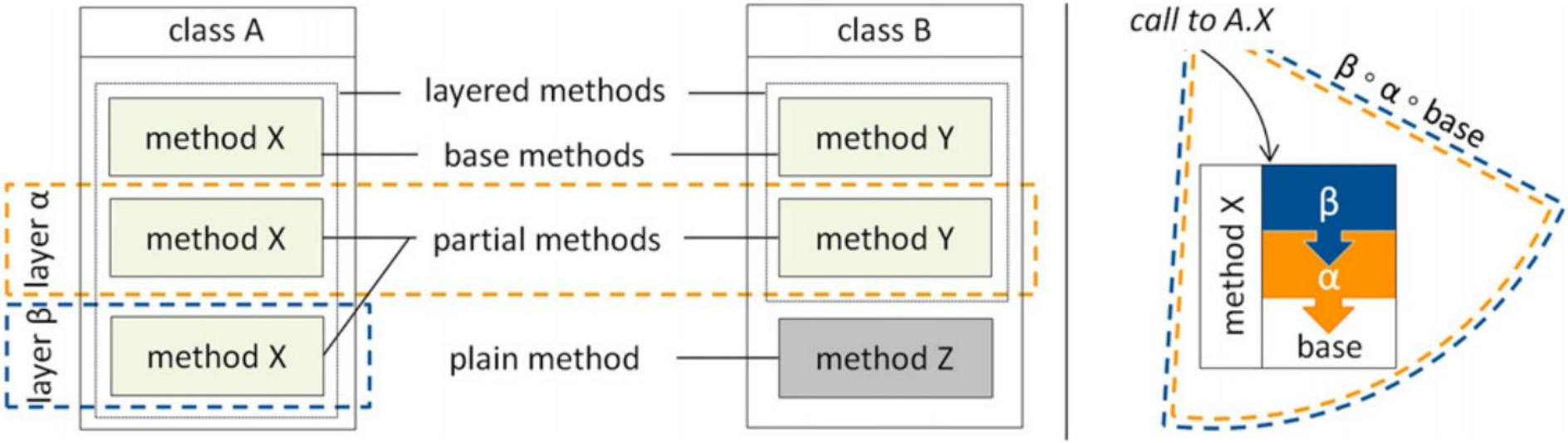
COP: ContextJS 2.0

Jakob Reschke, Marianne Thieffry

Web-based Development Environments
Hasso Plattner Institute, Software Architecture Group
Summer Term 2016, 13.07.2016

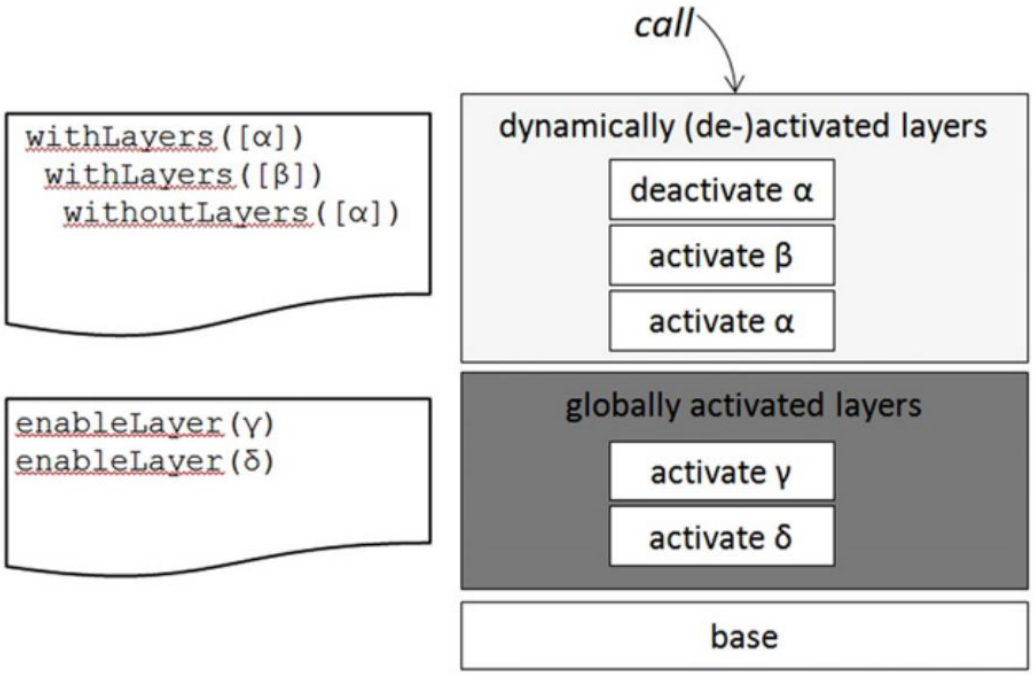
Demo
(a. k. a. What is COP?)

COP



An open implementation for context-oriented layer composition in ContextJS (Jens Lincke, Malte Appeltauer, Bastian Steinert, Robert Hirschfeld)

COP



An open implementation for context-oriented layer composition in ContextJS (Jens Lincke, Malte Appeltauer, Bastian Steinert, Robert Hirschfeld)

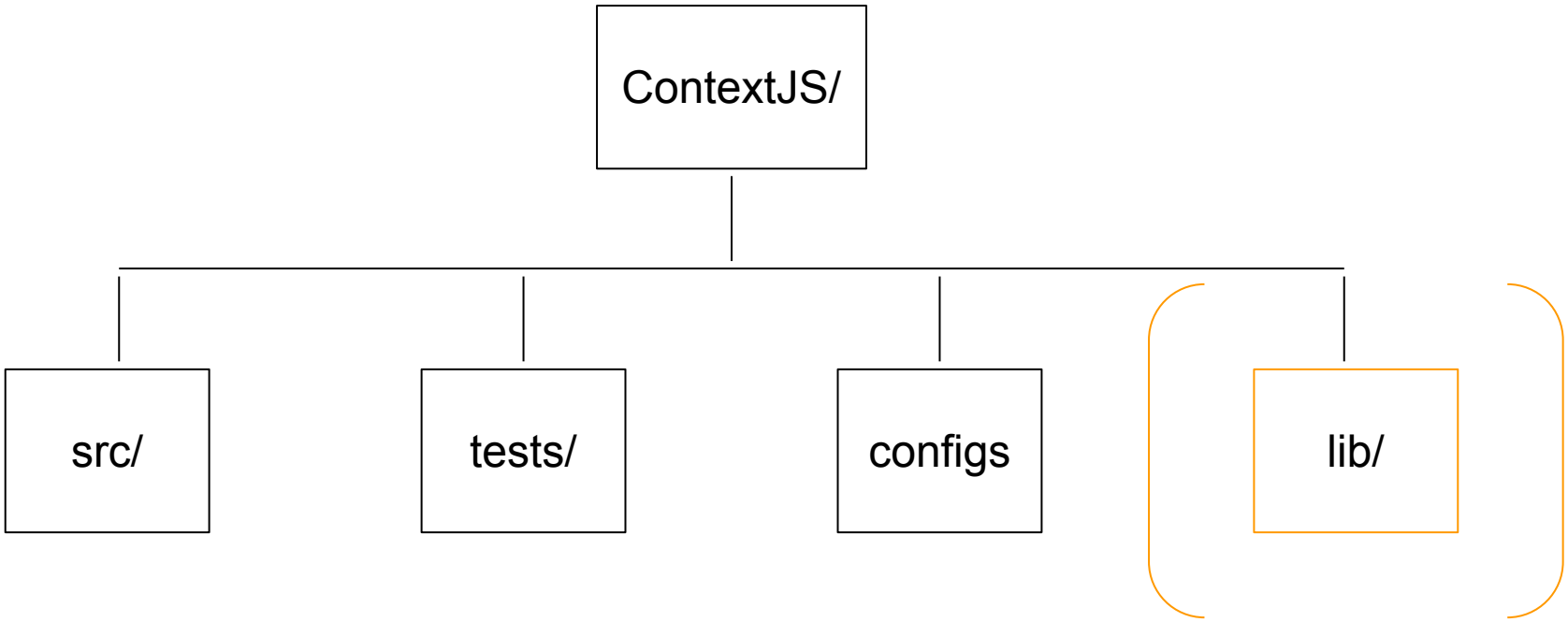
Old ContextJS

- Lively3 Module
- Uses Lively3/Prototype.js features
  - Object.subclass
  - “Global” object
  - ...
- Global “cop” object with COP operations
  - createLayer
  - withLayers
  - withoutLayers
  - a Trait for structural layer composition

New ContextJS

- **Standalone node module** without prod dependencies
- Using **ES6** instead of Lively Classes
- **“Slim” version** available for fast and easy use

Repository Structure

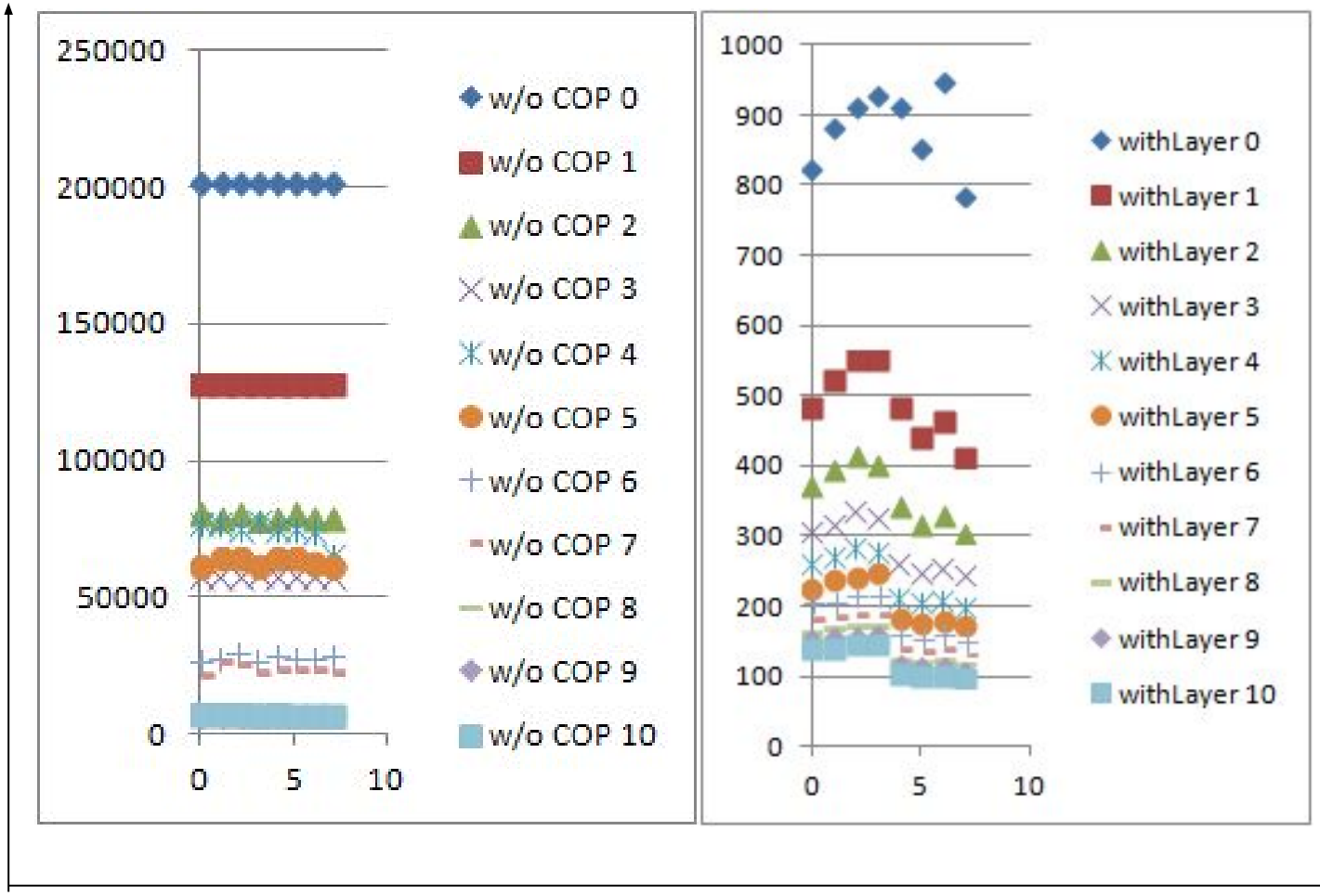


Changes and Challenges

- ES6 Classes = ES5 Constructors
- cop.createLayer → cop.layer
- ES6 (and node) condemn namespace pollution
- ES6 transpilation
- NPM prepublish & git repo
- Reliable chai import (node, karma, Lively) for tests

Benchmark Studies

Method invocations per second
(median of 30 samples for each revision)



Benchmark Code

Note the vertical scale!

Notable closed issues

- Removed references to lively.lang and prototype.js
- Refactored code to ES6
- Test cases ported to Mocha/Chai
- Tests run on TravisCI
- Publish ready package.json and bower.json files
- Lively4-friendly “persistently” layered methods
- Dropped deprecated syntax and features

Demo: ContextJS in Lively4

Let's make a bubble chart with d3 like this one:
http://bl.ocks.org/mbostock/4063269
in the lively4-cop-testingground

npm publish contextjs

Outlook

Wait for further feedback from real-world applications

Performance optimization

Appendix

Benchmark Code

Back to charts slide

```
class BenchClass {
  constructor() {
      this.counter_00=0;
      this.counter_01=0;
      this.counter_02=0;
  }
  countWithoutCOP(context){
      if(context.layer1) { this.counter_01++ }
      if(context.layer2) { this.counter_02++ }
  }
  countWithLayers() { this.counter_00++ }
}
```

```
Layer1.refineClass(BenchClass, {
    countWithLayers() {
        this.counter_01++;
        proceed()
    }
})

Layer2.refineClass(BenchClass, {
    countWithLayers() {
        this.counter_02++;
        proceed()
    }
})
```

Benchmark Code

Back to charts slide

```
function benchmarkWithContext(name, context) {
     benchmarkBlock(name, 16, function(size, obj) {
          for (var i = 0; i < size; i++) {
               obj.countWithoutLayers(context);
               obj.countWithoutLayers(context);
               obj.countWithoutLayers(context);
               ...
          }
     })
};

benchmarkWithContext("w/o COP 0", {}),
benchmarkWithContext("w/o COP 1", {layer1: true})
benchmarkWithContext("w/o COP 2", {layer1: true, layer2: true})
```

Benchmark Code

Back to charts slide

```
withLayers([L1], function() {
     benchmarkBlock("ContextJS:Method:WithLayer:1", 16,
     function(size, obj) {
          for(var i = 0; i < size; i++) {
               obj.countWithLayers();
               obj.countWithLayers();
               obj.countWithLayers();
               ...
          }
     }
})
```

```
withLayers([L1, L2], function() {
     benchmarkBlock("ContextJS:Method:WithLayer:2", 16,
     function(size, obj) {
          for(var i = 0; i < size; i++) {
               obj.countWithLayers();
               obj.countWithLayers();
               obj.countWithLayers();
               ...
          }
     }
})
```

Benchmark Code

Back to charts slide

```
function benchmarkBlock(name, unrolledOps, func) {
    var MAXSIZE = 1000000000;
    var TARGETTIME = 25;
    var time = 0.0;
    var size = 100;
    var ops = 0;
    var obj = new BenchClass();
    // warmup
    func(100, obj);
    // find good size
    while(time < TARGETTIME && size < MAXSIZE) {
        func(1, obj);
        var time1 = new Date().getTime();
        func(size, obj);
        var time2 = new Date().getTime();
        time = time2 - time1;
        ops = unrolledOps * size;
        size *= 2;
    }
    // measure
    let sample = new Array(30);
    for (let i = 0; i < 30; i++) {
        let time1 = new Date().getTime();
        func(size, obj);
        let time2 = new Date().getTime();
        let timedelta = time2 - time1;
        sample[i] = ops / timedelta;
    }
    let result = [name, size, ops].concat(sample).join();
    printEachResult(result);
}
```

Outline

- Minidemo
- COP
  - “Old” ContextJS
  - “New” ContextJS
- Demo
- Publish ceremony!
- Benchmark studies
- Milestones
  - Notable issues closed
  - Outlook