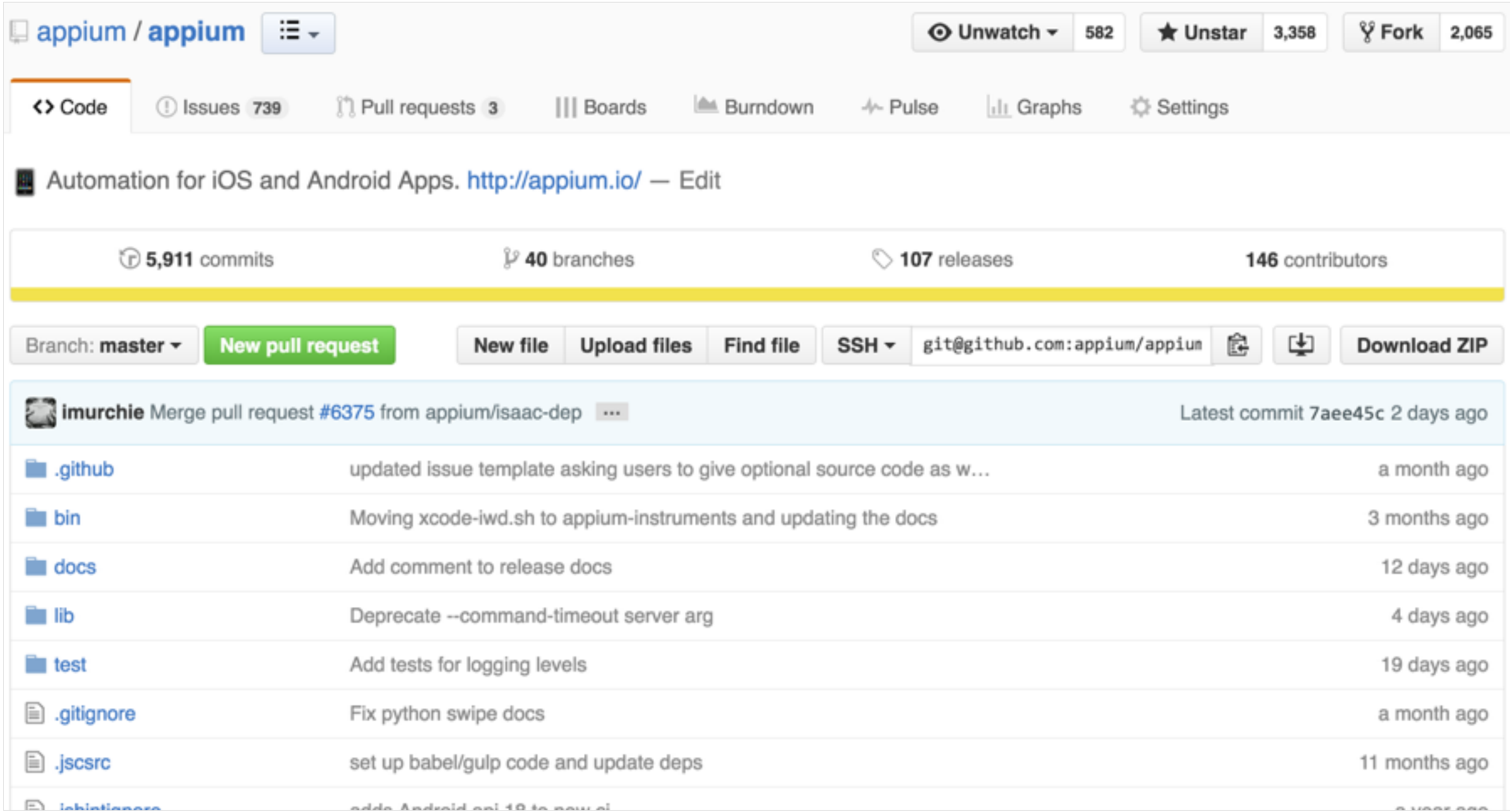

The Node Module Diaries
JONATHAN LIPPS, DIRECTOR OF ENGINEERING
@jlipps | @saucelabs | @AppiumDevs
April 12, 2016 | PhillyETE | Philadelphia, PA
SAUCELABS


appium
sponsored by Sauce Labs


Woven Automation Demo
localhost:5002
Full code available at https://github.com/saucelabs/appium-demos/python
#
from random import randint
from test_model import woven_app_model
opts = globals()
def automate_woven_app():
app = woven_app_model(opts['APP_PATH'], opts['APPIUM_PORT'],
opts['APP_PKG'], opts['APP_ACT'], opts['APP_WAIT'],
opts['SAUCE'])
try:
app.sign_in_button().click()
app.login(opts['EMAIL'], opts['PASSWORD'])
app.album("jlipps's Flickr").click()
for x in range(0, randint(2, 4)):
# scroll down a random number of times
app.scroll_down_photos()
# pick a random photo to look at
Carrier
3:10 PM
Woven
Safari
appium.sh (bash)
appium.sh (bash)
runner.py (python2.7)
info: Using locally extracted app: /var/folders/kt/6t1qwk1n4sq_3b5x4tly__5h0000gn/T/11414-22977-11flw3r/WovenIOS.app
info: Creating new appium session 5f9fd730-e8a3-477e-8ff8-052ce832cd23
info: Removing any remaining instruments sockets
info: Cleaned up instruments socket /tmp/instruments_sock
info: Cleaning up any tracedirs
info: Cleaned up /Users/jlipps/Code/appium-jlipps/instrumentscli0.trace
info: Parsed app Localizable.strings
info: Not setting locale
info: Setting iOS and app preferences
info: Starting iOS 8.+ simulator log capture
info: User specified default device, letting instruments launch it
info: Parsed app Info.plist
info: Wrote new app Info.plist with device type
debug: No device id or app, not installing to real device.
debug: Creating instruments
info: Instruments is at: /Applications/Xcode-6.1.app/Contents/Developer/usr/bin/instruments
info: [INSTSERVER] Instruments socket server started at /tmp/instruments_sock
info: Spawning Instruments with command: /Applications/Xcode-6.1.app/Contents/Developer/usr/bin/instruments -t /Applications/Xcode-6.1.app/Contents/Applications/Instruments.app/Contents/PlugIns/AutomationInstrument.bundle/Contents/Resources/Automation.tracetemplate /var/folders/kt/6t1qwk1n4sq_3b5x4tly__5h0000gn/T/11414-22977-11flw3r/WovenIOS.app -e UIASCRIPT /Users/jlipps/Code/appium-jlipps/lib/devices/ios/uiauto/bootstrap.js -e UIARESULTSPATH /tmp/appium-instruments/
info: And extra without-delay env: {"DYLD_INSERT_LIBRARIES":"/Users/jlipps/Code/appium-jlipps/build/Debug-iphonesimulator/libInstrumentsShim.dylib","LIB_PATH":"/Users/jlipps/Code/appium-jlipps/build/Debug-iphonesimulator"}
info: And launch timeout: 90000ms


appium / appium
Unwatch 582
Unstar 3,358
Fork 2,065
Code
Issues 739
Pull requests 3
Boards
Burndown
Pulse
Graphs
Settings
Automation for iOS and Android Apps. http://appium.io/ — Edit
5,911 commits
40 branches
107 releases
146 contributors
Branch: master
New pull request
New file
Upload files
Find file
SSH
git@github.com:appium/appium
Download ZIP
imurchie Merge pull request #6375 from appium/isaac-dep
Latest commit 7aee45c 2 days ago
.github updated issue template asking users to give optional source code as w… a month ago
bin Moving xcode-iwd.sh to appium-instruments and updating the docs 3 months ago
docs Add comment to release docs 12 days ago
lib Deprecate --command-timeout server arg 4 days ago
test Add tests for logging levels 19 days ago
.gitignore Fix python swipe docs a month ago
.jscsrc set up babel/gulp code and update deps 11 months ago


SAUCELABS
Testing at the speed of awesome.

Once Upon a Time...


appium
sponsored by Sauce Labs

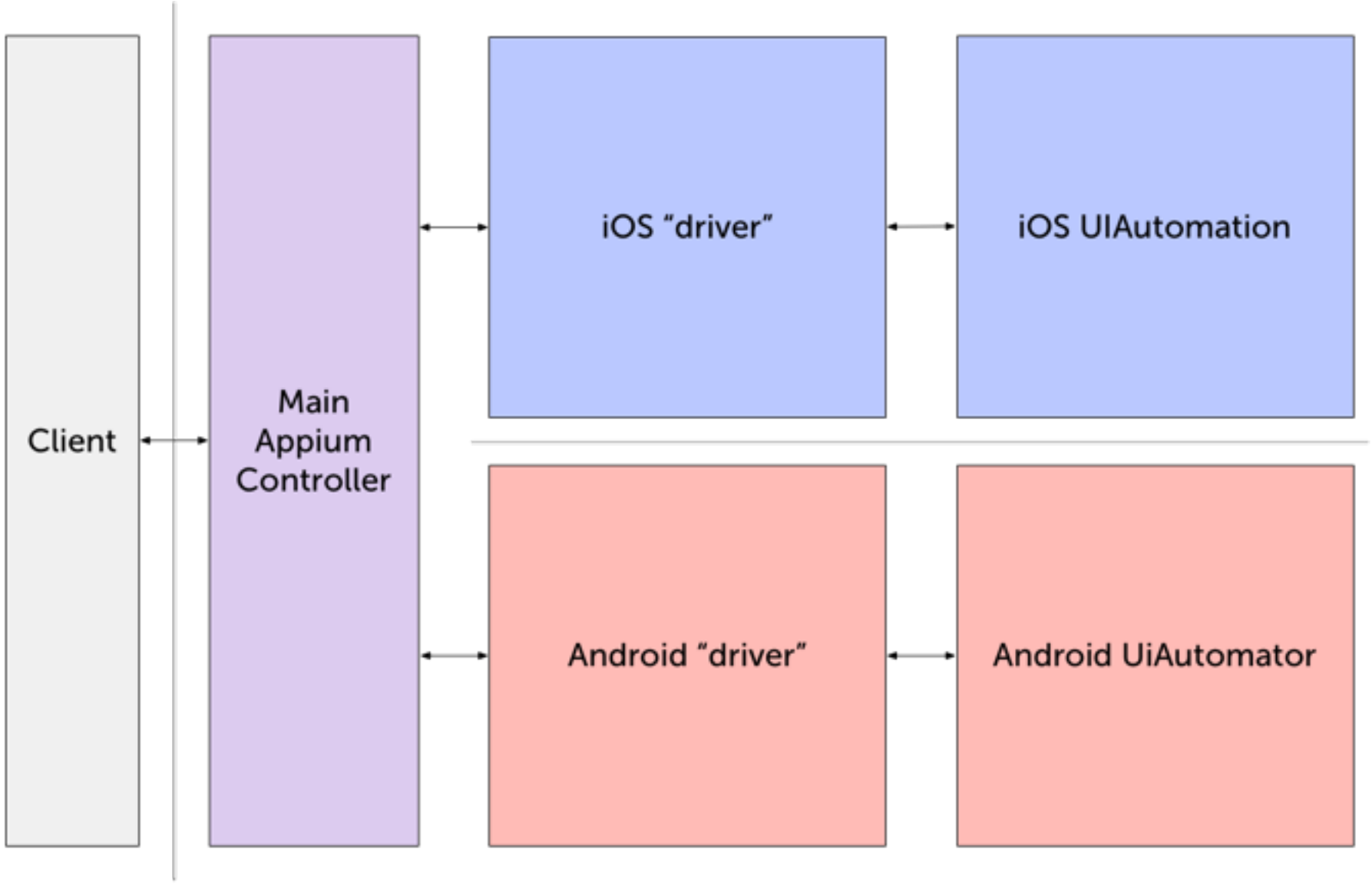

node
JS
TM

PROBLEMS

WHAT HAPPENS WHEN YOU TRY TO PATCH UP ALL THE BROKEN THINGS IN MOBILE AUTOMATION?

- Code not modular enough
- Code too modular
- ES5 hates new contributors
- CALLBACKS and async confusion
- Subprocess management
- Tight coupling between request and business logic
- Error handing all over the place
- OSS-specific project needs

MODULARITY?


Client
Main
Appium
Controller
iOS "driver"
iOS UIAutomation
Android "driver"
Android UiAutomator

CALLBACKS?

```
function foo (cb) {
    bar(function (err) {
        if (err) return cb(err);
        baz(function (val, err) {
            if (err) return cb(err);
            async.eachSeries(list, function (i, cb) {
                if (i % 2 === 0) {
                    fun(i, function (err) {
                        if (err) return cb(err);
                        otherFun(i, function (newVal, err) {
                            if (err) return cb(err);
                            val += newVal;
                        })
                    })
                }
            }, function (err) {
                if (err) return cb(err);
                cb(val);
            })
        })
    })
}
```

SUBPROCESS MANAGEMENT?

```
var spawn = require('child_process').spawn;

function boredTail (filePath, boredAfter, cb) {
    cb = _.once(cb);
    if (!boredAfter) boredAfter = 10000;
    try {
        var proc = spawn('tail', ['-f', filePath]);
    } catch (e) {
        return cb(e);
    }
    proc.on('error', function (err) {
        cb(err);
    });
    proc.stdout.on('data', function (chunk) {
        console.log('STDOUT: ' + chunk.toString('utf8').trim());
    });
    proc.on('exit', function () {
        cb();
    });
    setTimeout(boredAfter, function () {
        proc.kill();
    });
}
```

SOLUTIONS

REWRITE ALL THE THINGS!

- ES5 -> ES2015
  - `for … of`
  - arrow functions
  - real ‘classes’
- Callbacks -> `async/await`
- `child_process` -> https://github.com/appium/teen_process
- Pattern standardization and consolidation in helper libraries
- Proper separation of concerns / total rearchitecture
- … and many small modules!!! (“micromodules”)

CALLBACKS -> ASYNC/AWAIT

```
function foo (cb) {
    bar(function (err) {
        if (err) return cb(err);
        baz(function (val, err) {
            if (err) return cb(err);
            async.eachSeries(list, function (i, cb) {
                if (i % 2 === 0) {
                    fun(i, function (err) {
                        if (err) return cb(err);
                        otherFun(i, function (newVal, err) {
                            if (err) return cb(err);
                            val += newVal;
                        })
                    })
                }
            }, function (err) {
                if (err) return cb(err);
                cb(val);
            })
        })
    })
}
```

```
async function foo () {
    await bar();
    let val = await baz();
    for (let i of list) {
        if (i % 2 === 0) {
            await fun(i);
        }
        val += await otherFun(i);
    }
    return val;
}
```

child_process -> teen_process

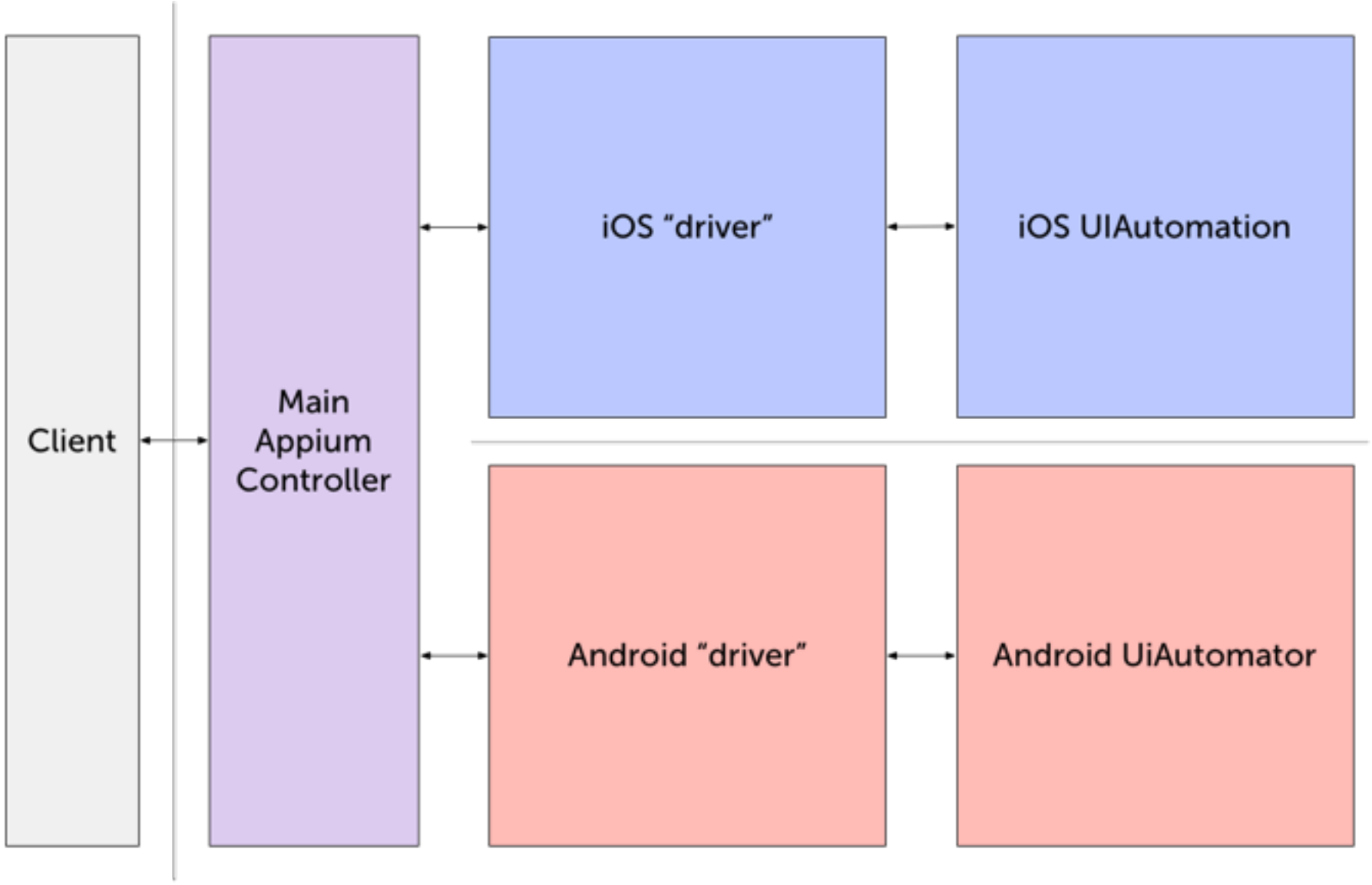
```
var spawn = require('child_process').spawn;

function boredTail (filePath, boredAfter, cb) {
    cb = _.once(cb);
    if (!boredAfter) boredAfter = 10000;
    try {
        var proc = spawn('tail', ['-f', filePath]);
    } catch (e) {
        return cb(e);
    }
    proc.on('error', function (err) {
        cb(err);
    });
    proc.stdout.on('data', function (chunk) {
        console.log('STDOUT: ' + chunk.toString('utf8').trim());
    });
    proc.on('exit', function () {
        cb();
    });
    setTimeout(boredAfter, function () {
        proc.kill();
    });
}
```

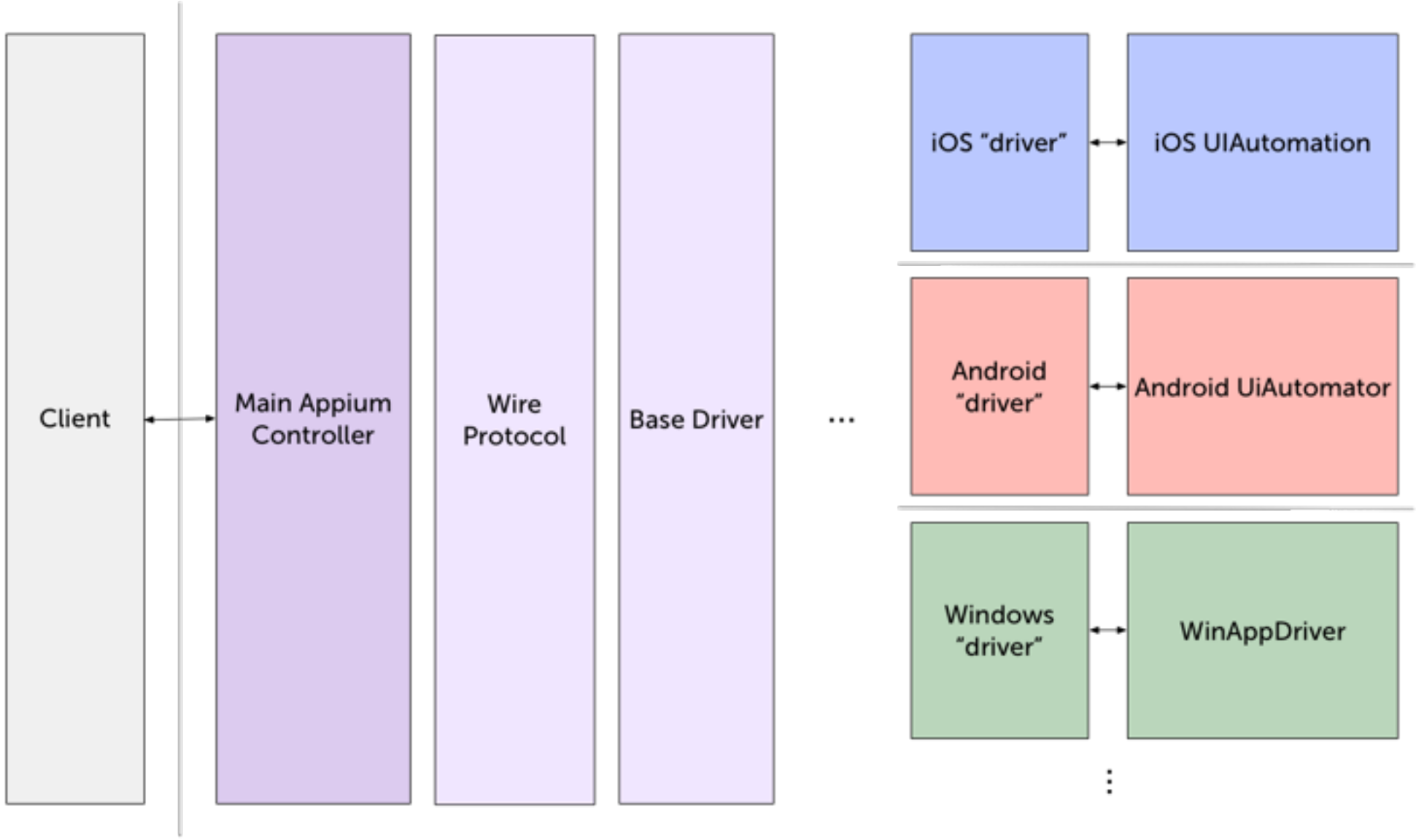
```
import { SubProcess } from 'teen_process';

async function boredTail (filePath, boredAfter = 10000) {
  let p = new SubProcess('tail', ['-f', filePath]);
  p.on('output', stdout => {
    if (stdout) {
      console.log(`STDOUT: ${stdout.trim()}`);
    }
  });
  await p.start();
  await Bluebird.delay(boredAfter);
  await p.stop();
}
```

MODULARITY / SEPARATION



MODULARITY / SEPARATION

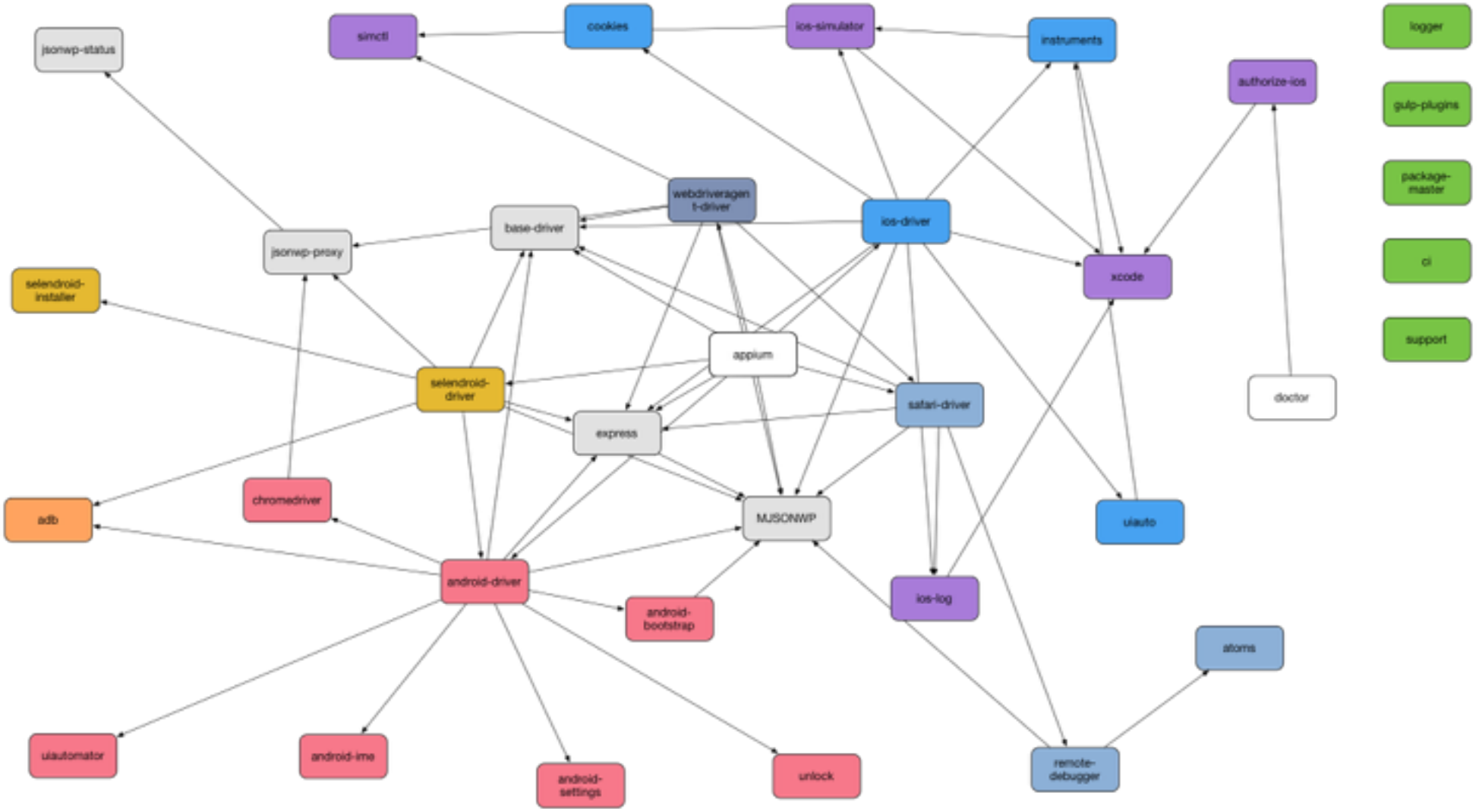


CONVENTIONAL WISDOM

OR ME ATTENDING TOO MANY CONVENTIONS?

- Micromodules / Tiny modules everywhere
  - One module per discrete bit of functionality
  - NPM and SemVer to manage module relationships
  - One-to-one relationship between modules and repos
- Microservices / SOA!
  - Monoliths are bad
  - Components should be hermetically compartmentalized and independent
  - Components should talk over the network using messaging or RPC

MICROMODULES!!1




disneyscreencaps.com

BENEFITS

WHAT DID WE THINK MICROMODULES WOULD GIVE US?

- Separation of concerns
- Increased test surface area and asynchronous CI for components
- Ability to version components separately and use NPM + SemVer for easy dependency management
- Ability to release components as their own modules for independent use by third parties

DRAWBACKS

WHAT ACTUALLY HAPPENED?

- Working with many repos is frustrating
  - Git history, repo stats, etc... really clunky to find
  - Context-switching when you need to work on a different module/repo
  - Git history clogged with “update this dep to version xx.yy.zz” and takes discipline to include context
- Working with many modules is frustrating
  - When everything is babelified, just `rm -rf node_modules && npm install` takes forever.
  - Cross-package/multi-package changes are hard—PRs all over the place, code must be merged in certain orders
  - Lots of module-level boilerplate, even when abstracted (gulp, build tools, transpilation and test watching, etc...)

DRAWBACKS CONT...

WHAT ACTUALLY HAPPENED?

- Dependency management always sucks, and now we had a lot of it
  - 3rd party deps are always changing. Who wants to update `lodash` in 50 different places?
  - 3rd party deps at different versions across 1st party modules can cause problems
  - Even 1st party deps at different versions cause problems, e.g. with `instanceof` checks
- Per-module CI is nice but at the end of the day doesn't save time. Smart incremental builds are different than per-repo or per-module builds.

BENEFITS REVISITED

WHAT DID WE ACTUALLY NEED?

- Separation of concerns
- Increased test surface area and asynchronous CI for components
- ~~Ability to version components separately and use NPM + SemVer for easy dependency management~~
- ??? Ability to release components as their own modules for independent use by third parties

WHAT HAPPENED?

WHY DID WE OVERENGINEER?

- Assumed “micromodules” === “microrepos”
- Applied the “open source” model of many separate projects too eagerly. We are open source, but we’re still (mostly) one app
- Fooled by the family resemblance between “micromodules” and “microservices”
    - Contemplated a “microservice” architecture without understanding why it didn’t apply in our case
    - We had components that did RPC, but crucially ours was 1 app in memory, with subprocesses, that didn’t need to scale internal components horizontally. It already was a “service”.
- Superficially, it seemed like we needed these strategies. But we didn’t.

MICROSERVICES / SOA

WITHOUT THE CARGO CULT

- You need to distribute processing or data across many machines/VMs/containers because one isn't good enough
- Your app conceptually has multiple distinct responsibilities which can feasibly be broken up into different 'services'. You already have a good idea of where to draw these lines
- Your services can do their job behind load balancers / while horizontally scaled
- You need fault tolerance so that your services/containers/VMs can crash without affecting customer experience
- Changes usually don't happen across multiple services simultaneously
- All of the above are so important that you want to deal with a whole new level of complexity, asynchronicity, and orchestration

MICROMODULES

WITHOUT THE CARGO CULT

- You need each module to be `npm install`-able on its own
- You are OK religiously following SemVer
- Each module is conceivably genuinely useful as a standalone piece of software
- For 1st party (private or for-all-intents-and-purposes private) modules, you don't have single-leaf nodes.
    - If a module is imported by only one other module, does it really need to be published and managed separately?
- Micromodules philosophy is great for encouraging more READMEs, more tests, more CI, more separation of concern, etc... But it does not require those nor do they require it.

MICROREPOS

WITHOUT THE CARGO CULT

- You want different communities of people to engage with or manage different aspects of your project
- … and that's about it! You probably don't want microrepos.
- Seriously, check out Babel and other monorepos
- Remember, "monorepo" !== "monolith"

MICRO OR MONO

I am building…	Microservices	Micromodules	Microrepos
Distributed SaaS app, cloud platform, hugely popular web app, etc…	Yes	Yes - 1 per service - 1 per set of shared utilities	Maybe
Desktop app or web app with low scaling requirements	No	No	No
Open source library with lots of individually useful components	No	Yes - 1 per individually useful component	Probably not

HAPPY ACCIDENTS

BEING WRONG HAS ITS PEDAGOGICAL ADVANTAGES

- We learned a lot about NPM and module management
- We built useful tools to help us manage lots of modules (packageweb, turtledeps, diagnoss) and discovered others (greenkeeper, semantic release)
- The constraints of micromodules led us to spit up our app and modularize it across much better boundaries. We learned so much more about our problem space.
- We became experts in Node build tools and transpilation. The ES2015 rewrite has paid huge dividends, especially `async/await`
- We wrote many more tests and READMEs and have CI set up for everything

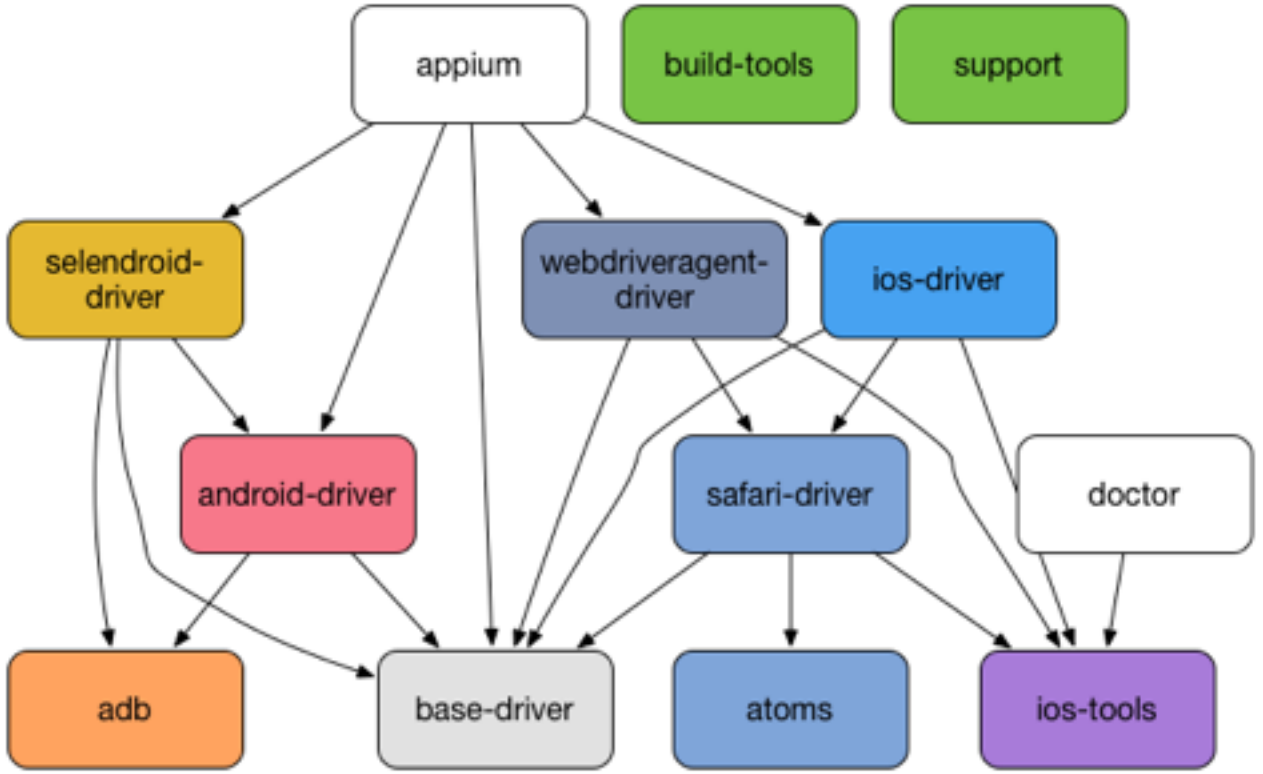
FINAL TAKEAWAYS

THE END OF OUR CAUTIONARY TALE

- The benefit of ES2015 and in particular `async/await` was totally worth the cost and complexity of transpilation
- Develop a strong cargo-cult radar
- Adopt a default stance of suspicion towards trends and buzzwords
- Before you can microservice or micromodule, you must deeply understand the responsibilities and capabilities of your app
- You really want a strong distinction between 1st party deps and 3rd party deps (left-pad, anyone?)
- Microservices/micromodules are no substitute for, and are not identical with, general good programming practices (modularity, separation of concerns, documentation and READMEs, writing testable code, writing tests, etc...)
- At the end of the day, it is only by diving in and building an architecture that you will come to understand whether it is the right one or not.

APPIUM'S FUTURE

A MORE MODEST PROPOSAL



RESOURCES

EXERCISES FOR THE READER

- Laurie Voss's talk on Microservices: https://www.youtube.com/watch?v=VijSWboZP-8
- Martin Fowler's Microservices article: http://martinfowler.com/articles/microservices.html
- Dan Luu on Monorepos: http://danluu.com/monorepo/
- Babel's Monorepo justification: https://github.com/babel/babel/blob/master/doc/design/monorepo.md
- Simon Stewart on Monorepos: http://blog.rocketpoweredjetpants.com/2012/11/ruminations-on-code-bases-i-have-known.html

The End

@jlipps | @saucelabs | @AppiumDevs

Questions?

@jlipps | @saucelabs | @AppiumDevs

Thank you

@jlipps | @saucelabs | @AppiumDevs