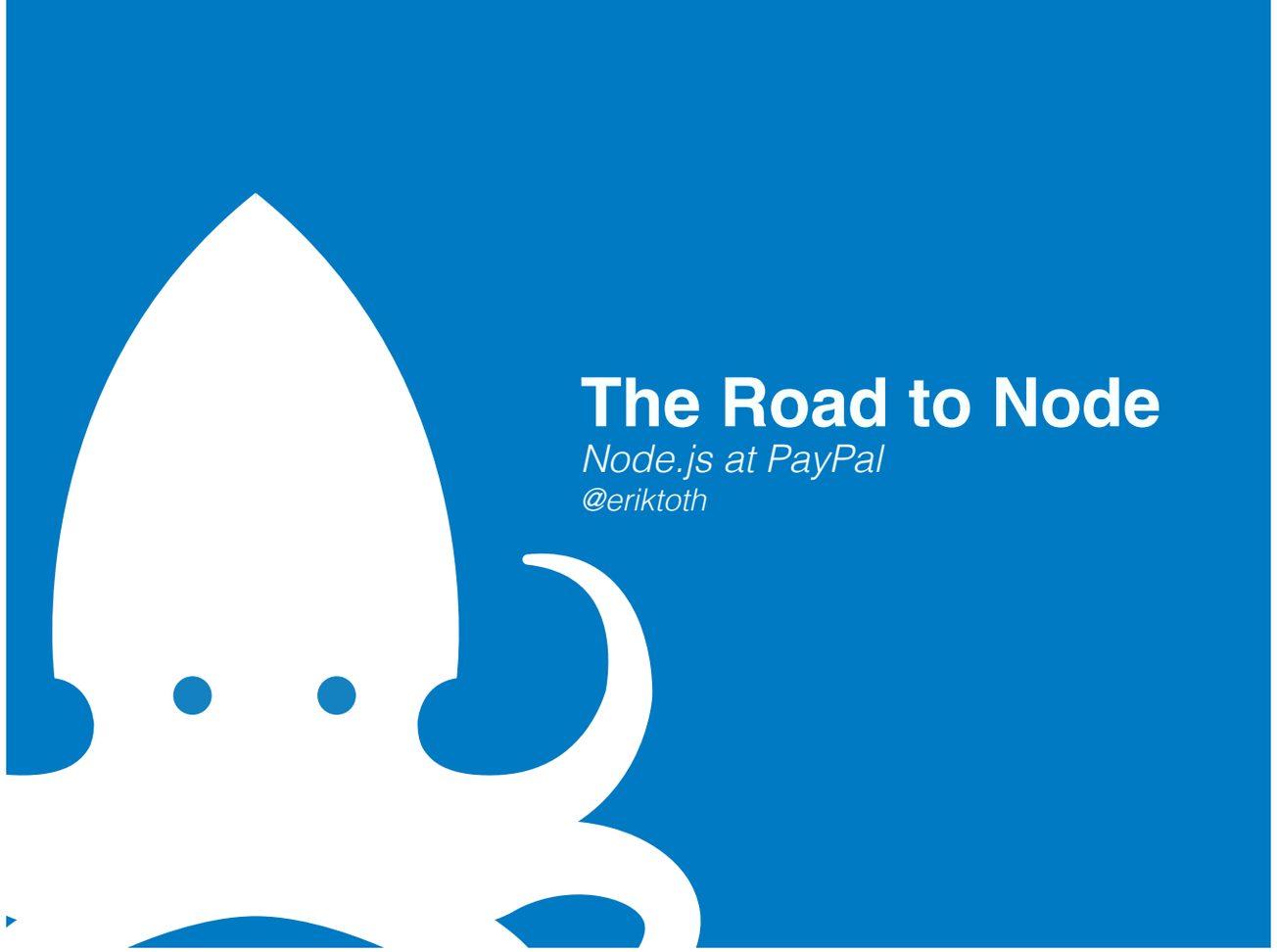


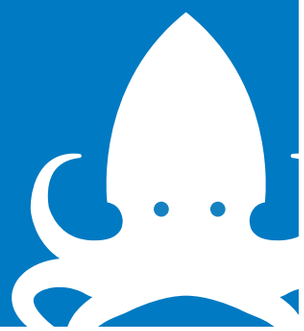
The Road to Node

Node.js at PayPal

@eriktoth

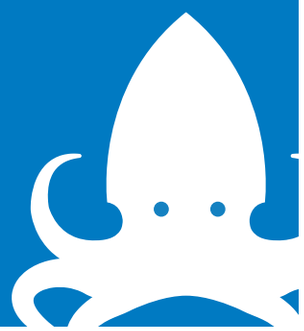
The Setup

A little context



Who am I?

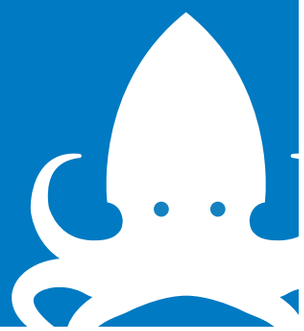
Where am I? Why am I here?



- Say your name :).
- Software Engineer working on Node.js infrastructure at PayPal
- Talk about what's been happening at PayPal and show some code.
- Sometimes non-technical solutions are the most important.

Who cares?

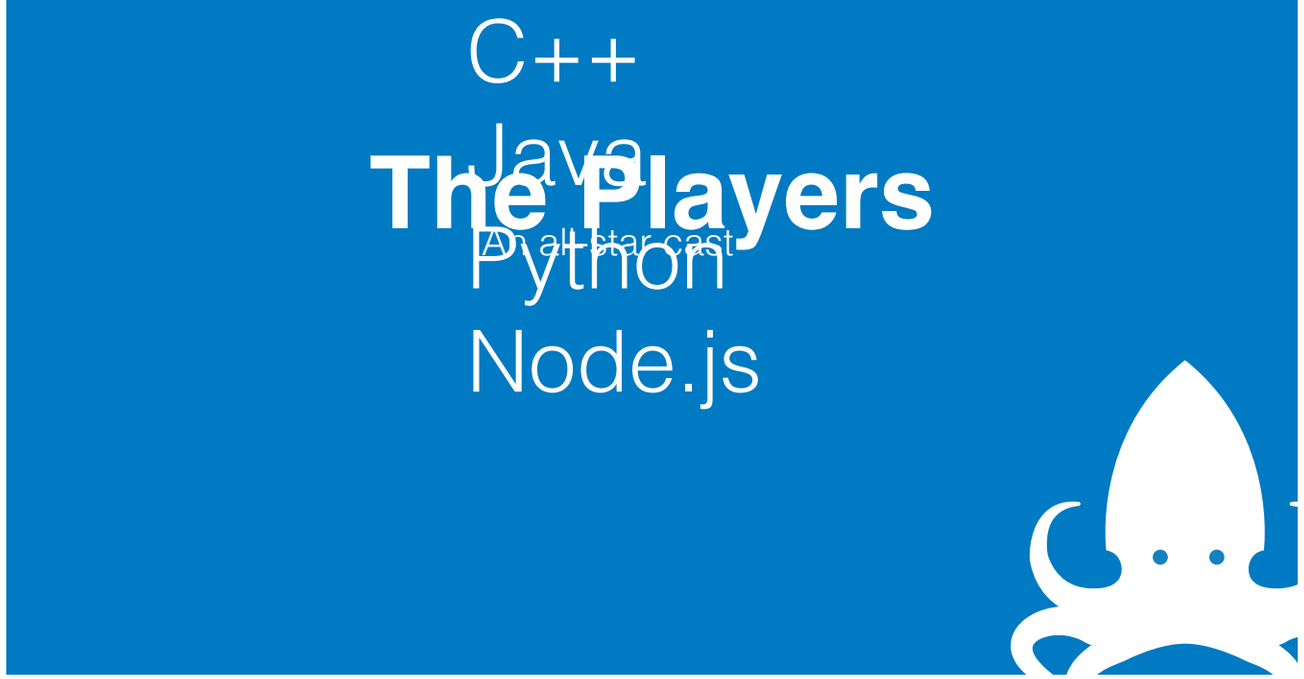
Maybe you. My boss. My boss' boss.



- Process + Technology are integral to affecting change at PayPal. "Institutional inertia."
- Velocity. Fail fast.
- Really, our learnings are in the context of the web application tier.
- Anyone interested in migrating to Node.js

The Players

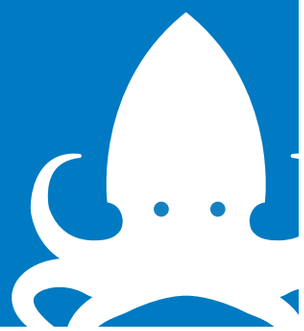
An all-star cast



- All started on C++, Java for years
- Python solved a lot of hard integration problems
- Node.js - the platform everyone loves to hate. Because JS is easier or problems were solved?

The Challenges

Let's set some ground rules



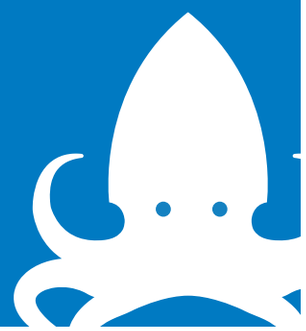
- Starting Point
- What were the goals? How will we know when we were done? What does success look like?

Thread.interrupt();

(Thread == PayPal for those following along at home.)

Some serious changes were needed:

- Higher change velocity in the front tier.
- More robust user testing (qual and quant).
- Improved developer productivity.



- We needed to stop some old practices and replace with new.
- Avoid technology vacuum! Don’t stop the world!

Q.defer();

Restraint (or procrastination
depending on who you ask.)

We had no intention of solving everything and really did our best to:

- stop ourselves from creating problems for the sake of solving them.
- find problems that weren't ours to solve (tricky, tricky).
- know our role and where Node.js fits into the bigger picture.
- stand on the shoulders of giants.

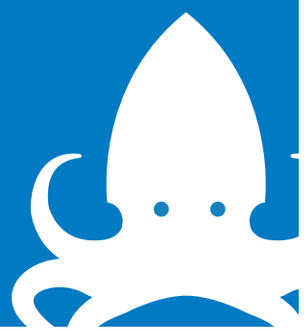
- Without restraint there's no end game.
- It's not an academic exercise, it's a business.
- Technology isn't the solution for everything, or even most things.
- Many problems were already solved, so why do so again.

transform: scale(2, 2)

(scaleX, scaleY, scaleZ, wait Z!?)

Multi-dimensional scaling:

- Runtime performance (obviously)
- Developer productivity
- Accommodate n developers/teams

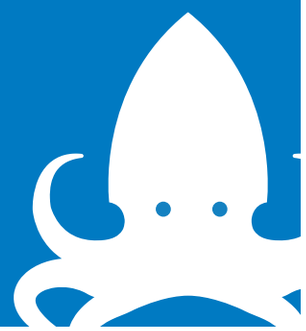


Scaling on many fronts

- Large engineering org.
- Lots of type of products.
- Avoid/obviate tribal knowledge.

The Process

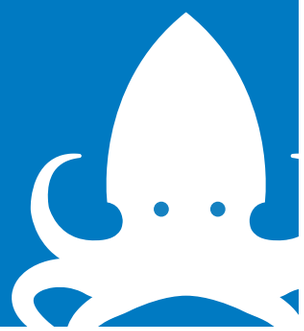
Can’t we get to code already?



cow.isSacred = false;

Kobe code?

- Open the code for all to see.
- Enterprise GitHub was instrumental.
- Operate internally as OSS teams work externally.



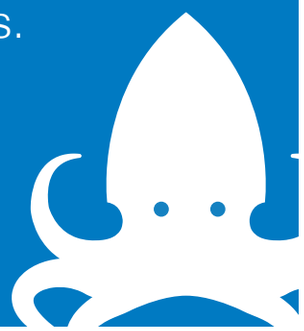
Share and share alike

- Turns out JS is great for transparency.
- CI/CD was huge side-benefit.

tasks.slice(-1);

// tasks = [“design”, “write tests”, “code”]

- Treat code written internally as if you’re going to publish it publicly.
- No passes or shortcuts because we’re “friends” and it’s “just for our own use.”
- Research similar solutions and patterns.



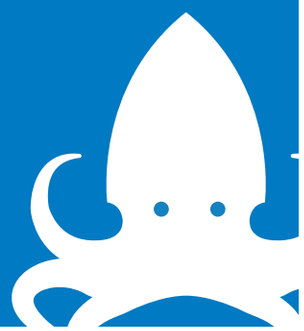
Meaningful abstraction.

- Each community has their own conventions. Don’t break them because your code is isolated.
- e.g. our promises contract.

completed.slice();

(Imitation is the sincerest form of flattery. So is copying.)

- Take advantage of work done by others.
- Fix where others fell short, but use the ideas that were successful. Don't "fix" what others did "wrong."

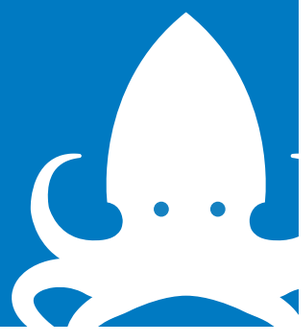


Build off other's successes.

- We had reference code in C++, Java, and Python
- Opinion vs fact.
- Our approach has been at the dismay of engineers. we strive to do less work. some engineers optimize for fewer keystrokes, but they should strive to delete code.

The Kraken

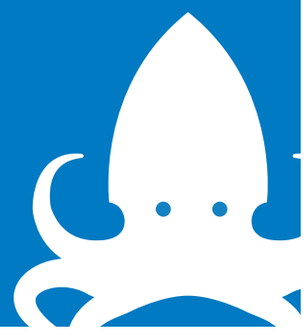
Something about “releasing” a thing?



kraken !== “framework”

(Don’t call it a comeback, either.)

Do call it a collection of modules, I suppose.



- Supports a somewhat specific use-case.

kraken-js API for creating express apps

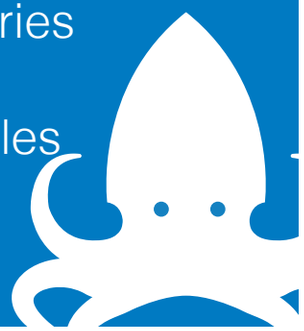
lusca appsec middleware for express

adaro shim to add request context to template resolution

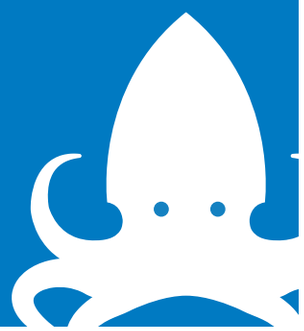
makara i18n support for Dust.js

kappa npm proxy for private registries

misc. additional supporting modules

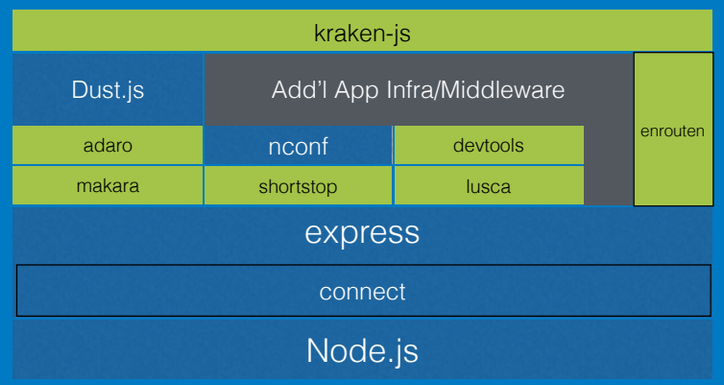


Codes
(finally.)



Okay, not yet.

Obligatory collection of miscellaneous boxes and whatnot.



- Diagrams not adequate.

Part 2

Revenge of the toy language

Back to Basics

What is Node.js exactly?

- v8 + core libraries + npm
- sweet spot: IO bound applications
- everything is a module
- a module is just a file (example?)

npm

The magic within

- packaged modules/reusable (no Maven/Ivy Hell)
- can use git or npm - kappa! 1) private npm 2)availability 3)security
- organizing large codebases
- anatomy of a module:
    - code - jshint
    - tests - tape
    - code coverage - istanbul
    - scripts - shebangs
- considerations: multi-platform support is not free

native modules

rough road ahead. vigilance is key.

- v8 changes affect compatibility
- platform support and compilation
- alternatives: interprocess communication/http/binary protocols

javascript on the server?!

I prefer to call it java'scrypt.

- functional programming: facts and fallacies
    - codebases *can* scale/incredibly expressive
- es5 goodness (no browser compatibility woes)
- control flow (callback hell?)
- higher order functions/lambdas !== async
- (call|err)backs vs promises
    - readability
    - composability
    - performance

“java” + “script”, right?

The migratory patterns of the modern engineer.

- It’s not java to node, it’s java to javascript
- embrace the language
- embrace the event loop
- embrace loose typing
- embrace functional programming

Thanks.

- @eriktoth