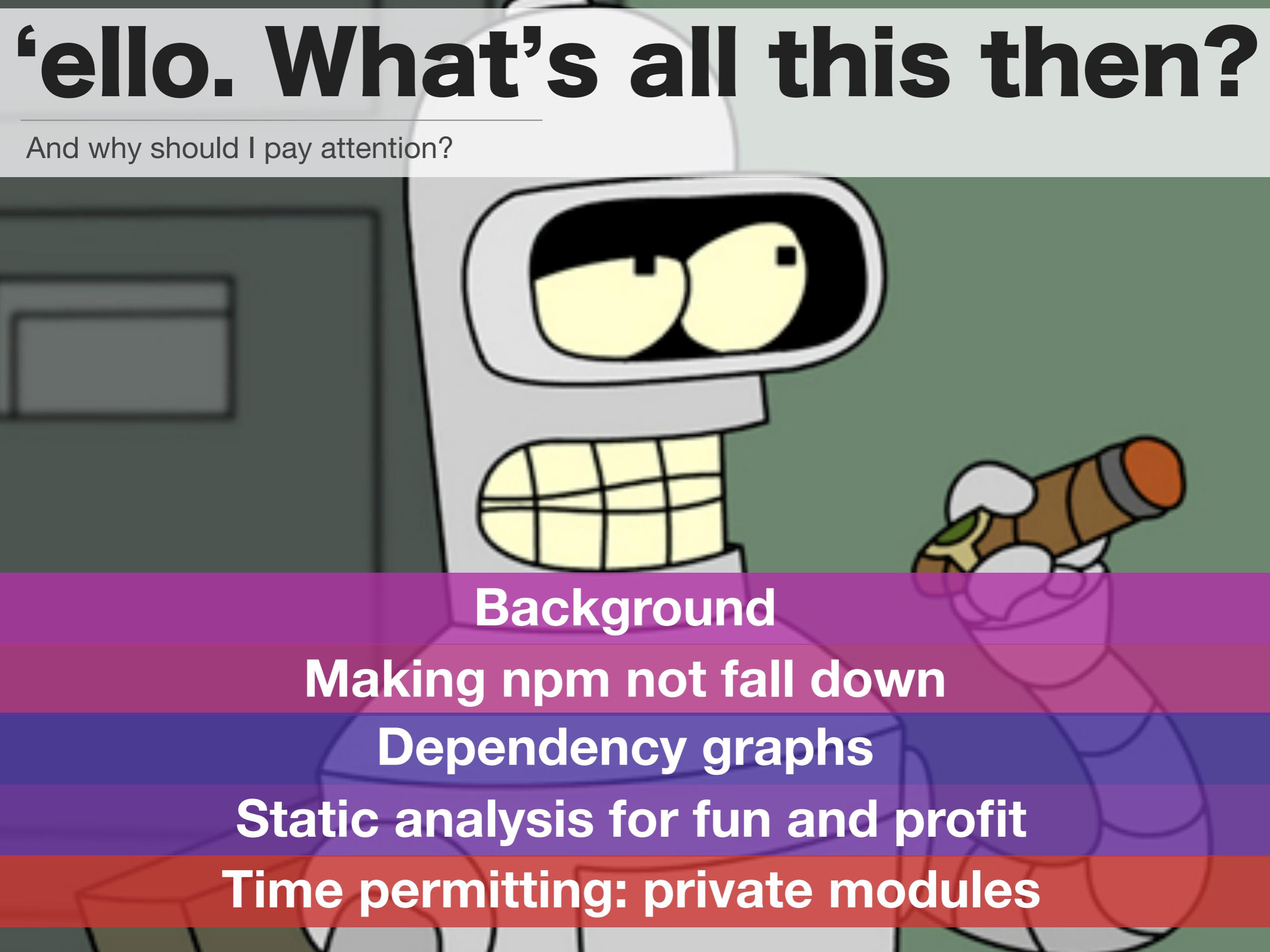


**THE FINAL  
FRONTIER**

# ‘ello. What’s all this then?

And why should I pay attention?

A cartoon illustration of a white robot head with large yellow eyes and a black mouth. It is mounted on a grey neck and has a yellow grid patterned torso. A speech bubble originates from its mouth. The background is a green wall with a dark grey door on the left. The robot is holding a brown and orange cylindrical device in its right hand.

Background

Making npm not fall down

Dependency graphs

Static analysis for fun and profit

Time permitting: private modules

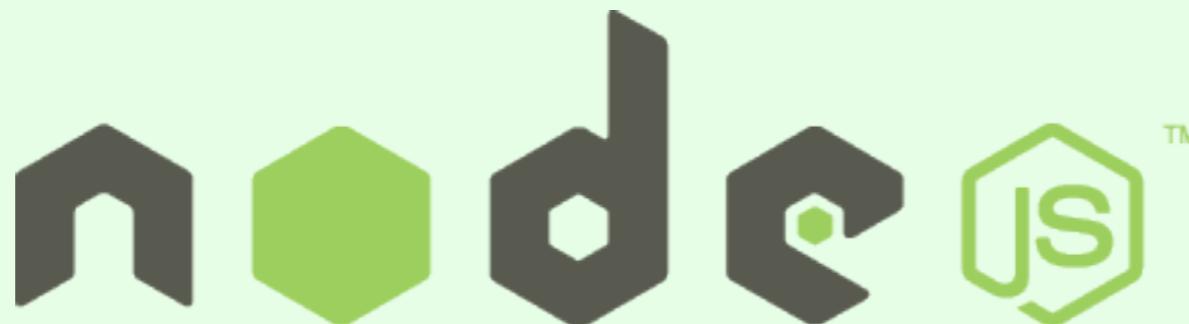
# Who are you exactly?

And why should I pay attention?

*Founder & CEO at*



*Actually has five (5) years of experience with:*



# Who are you exactly?

And why should I pay attention?

*In May 2013, Nodejitsu acquired*



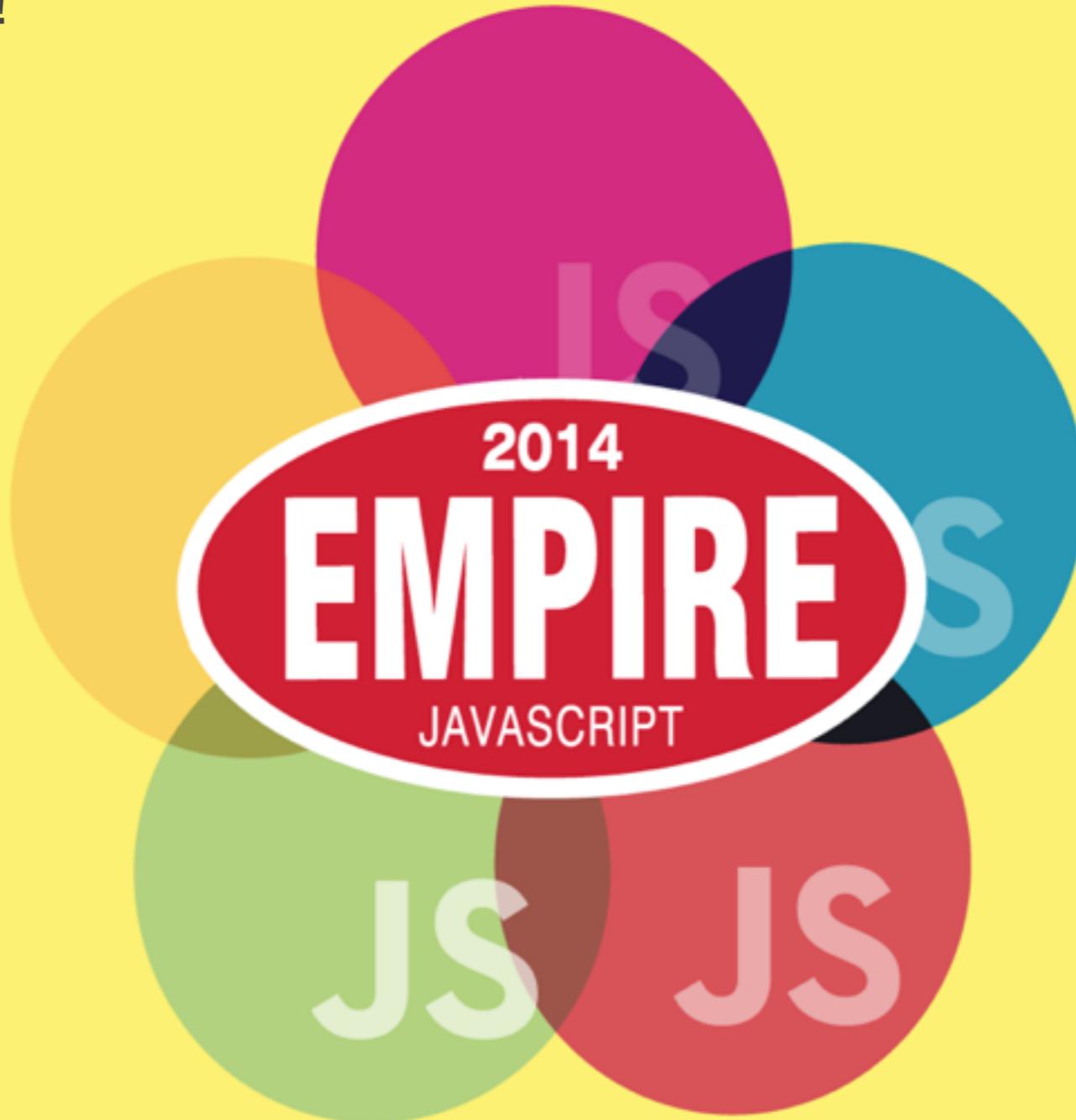
Iris Couch

*Until January 2013, IrisCouch ran the npm registry*



# Who are you exactly?

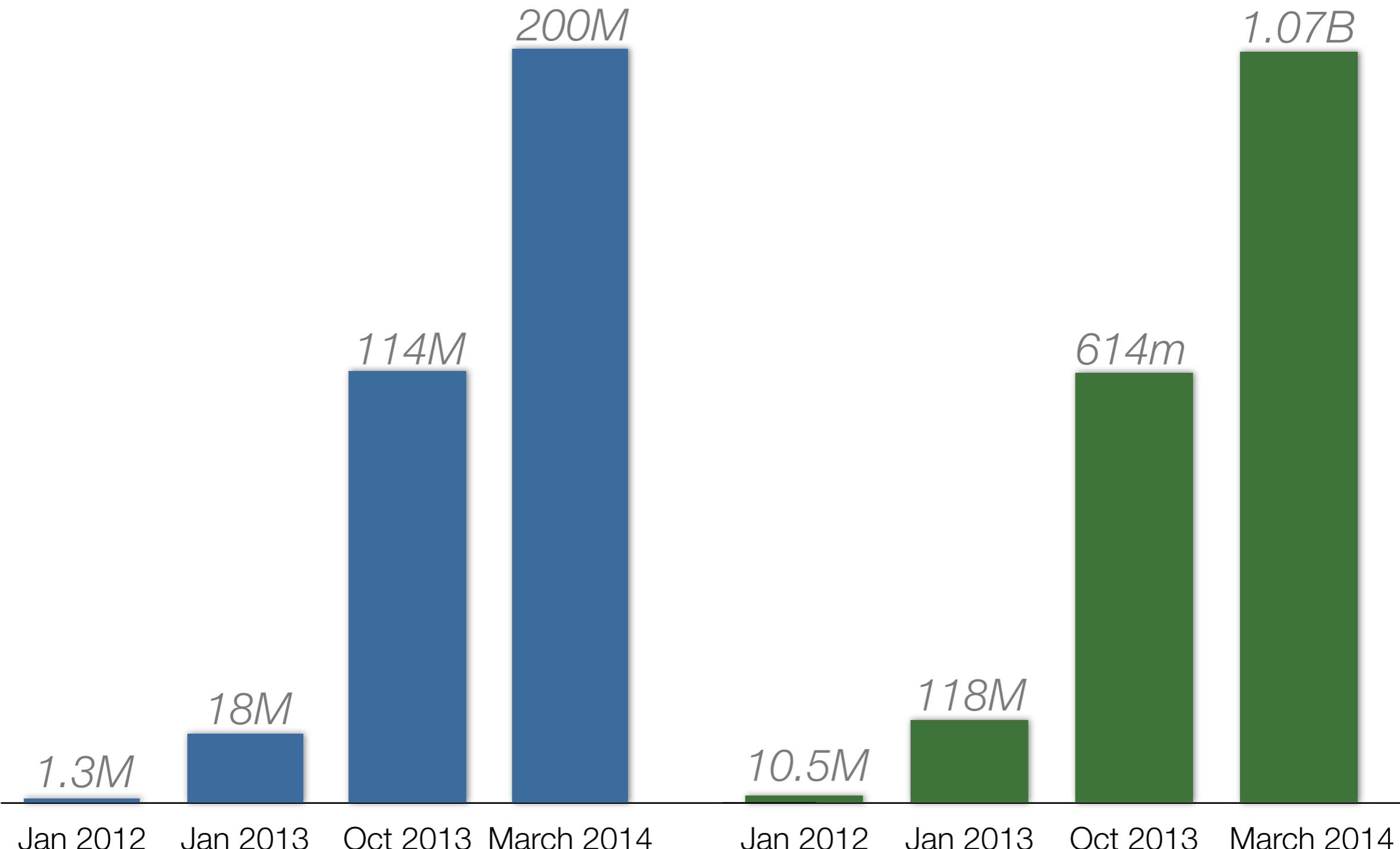
Come to EmpireJS!



<http://2014.empirejs.org>

# Why should I care about ?

Does two years of 1000% growth impress you?

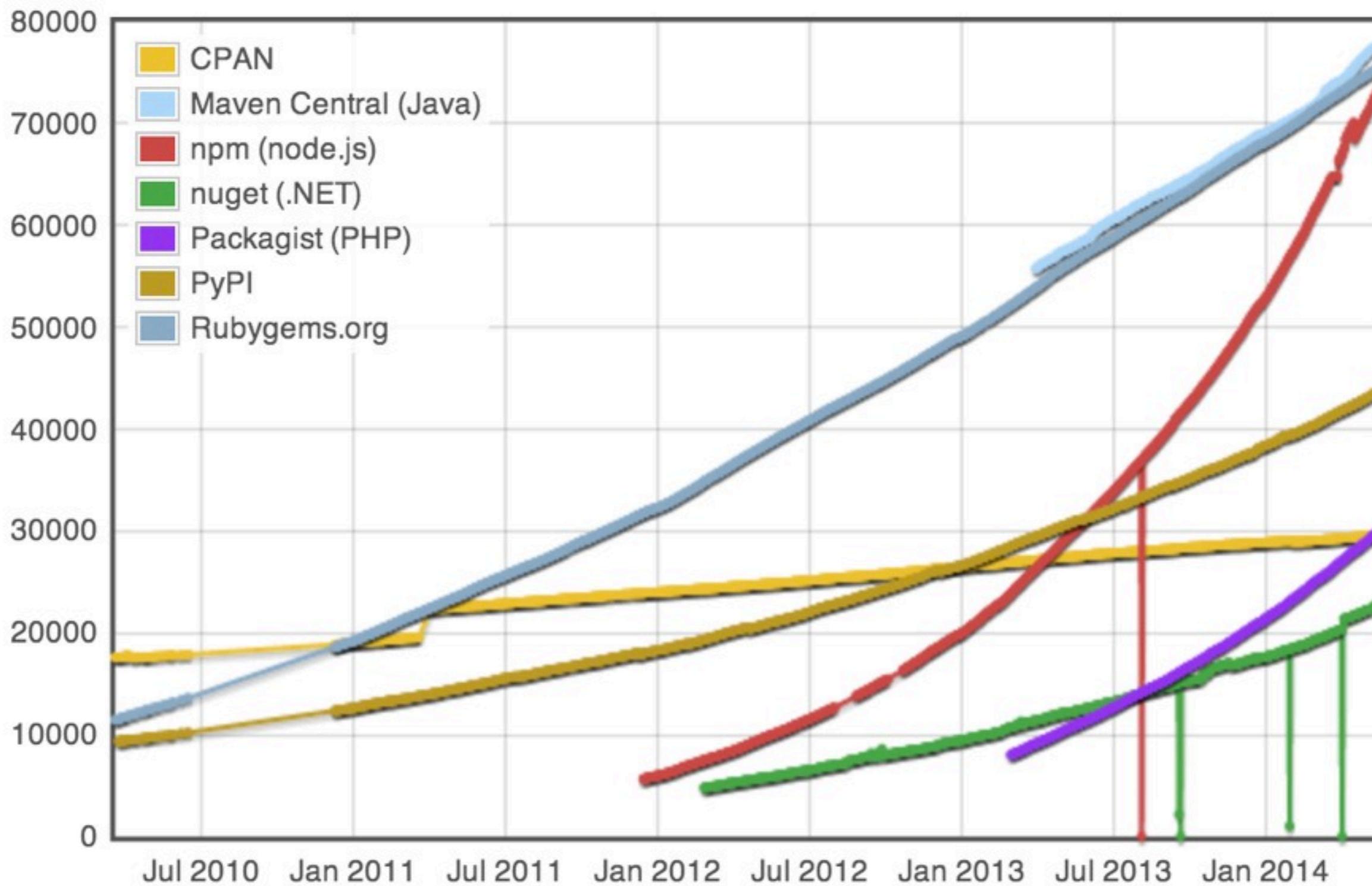


Module Downloads

Total Requests

# Why should I care about **npm** ?

Because it will soon be the largest Open Source platform worldwide



Source: [modulecounts.com](http://modulecounts.com)



CHESTER A. ARTHUR

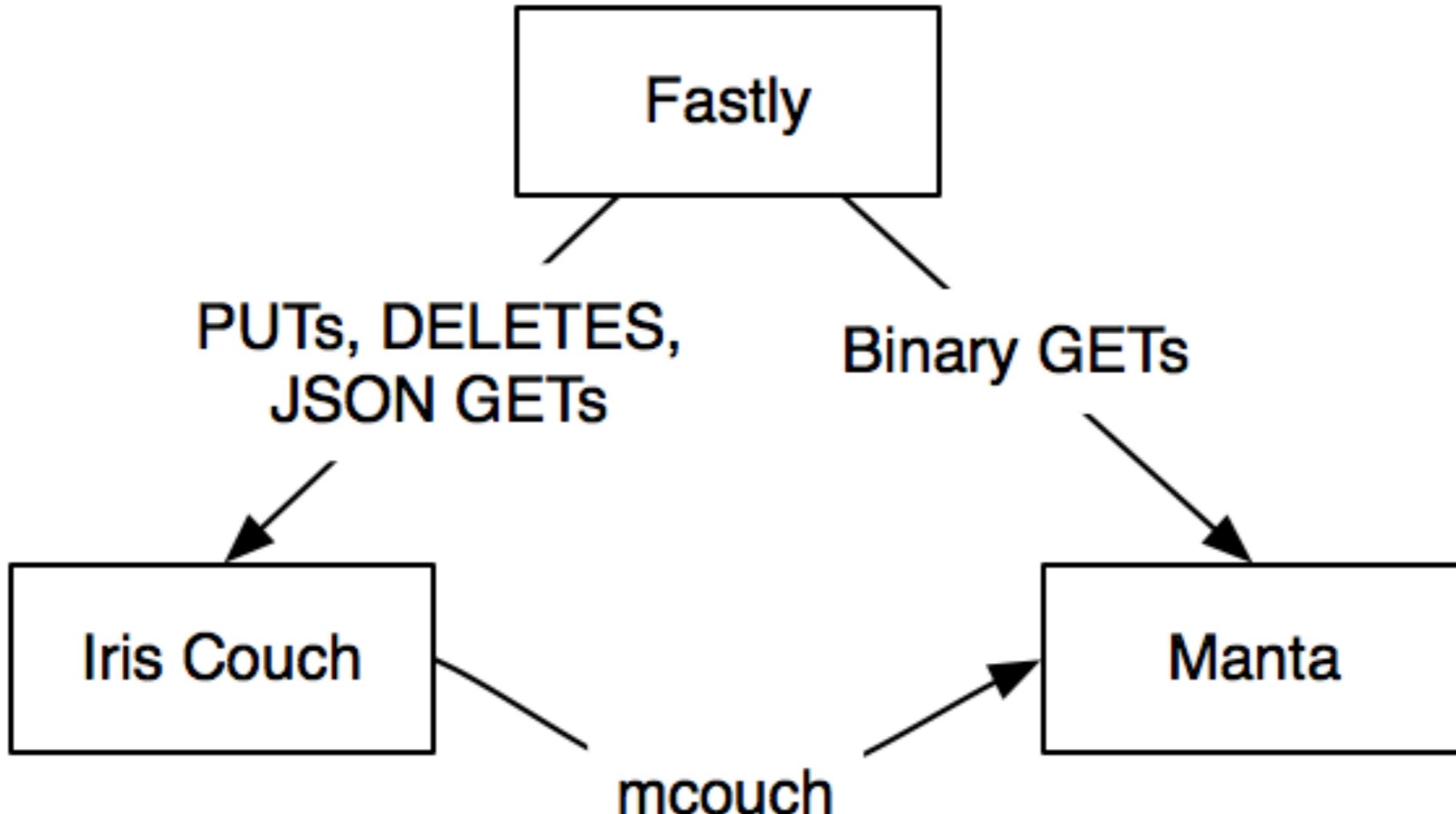
WAPUM

FALL  
DOWN



# Make not fall down.

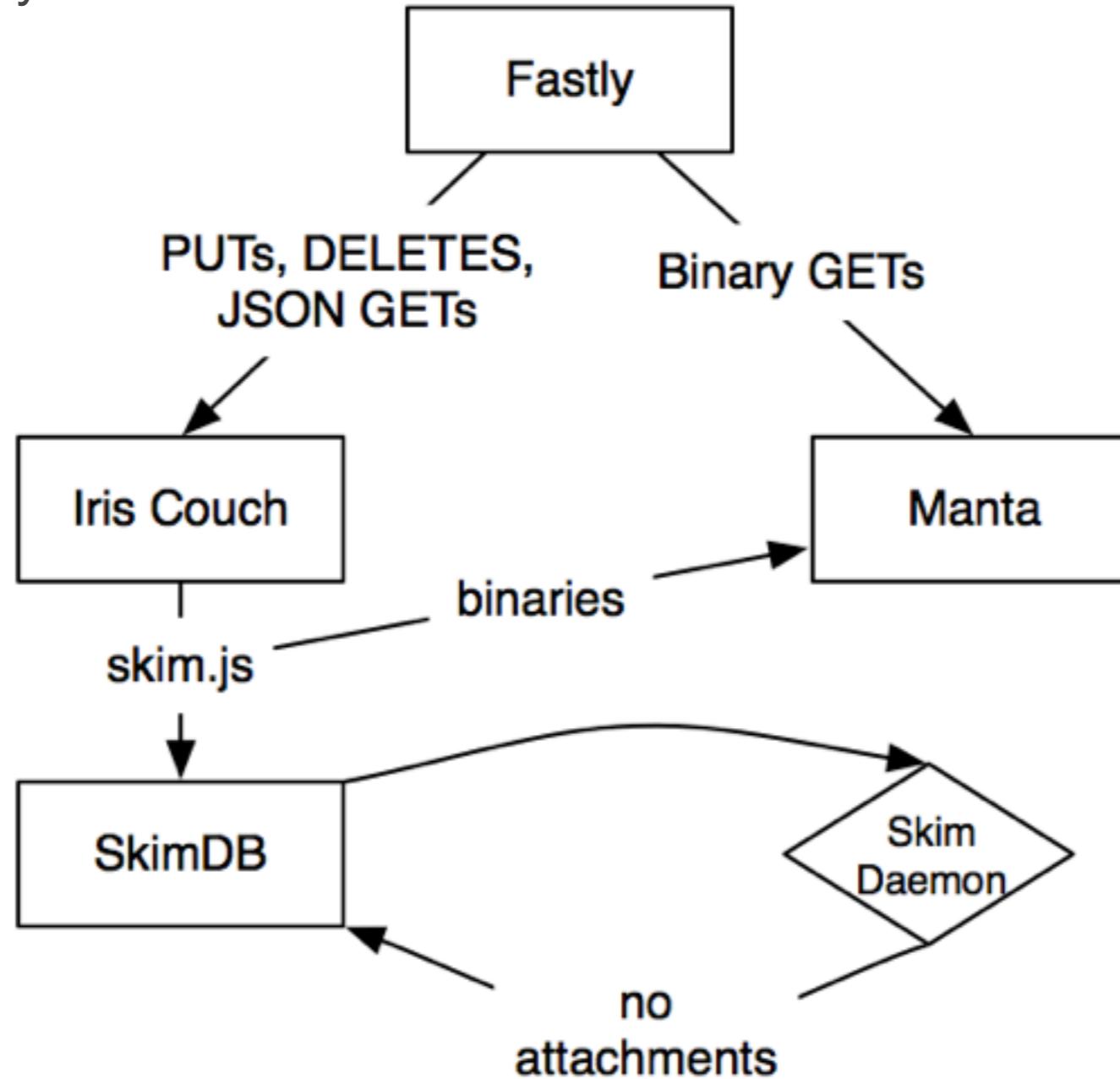
Make sure you've got your CAP on.



*January 2014*

# Make not fall down.

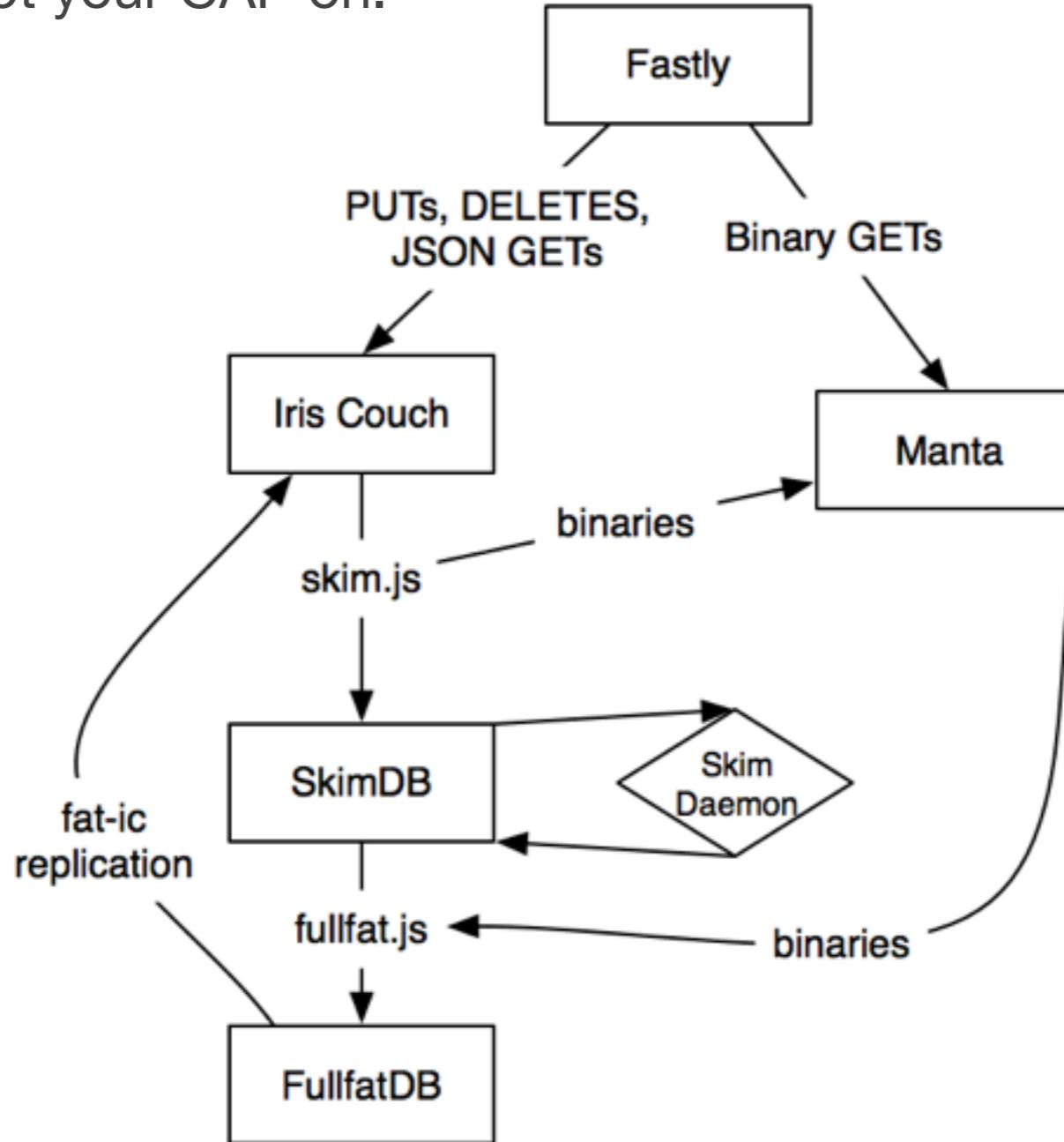
Make sure you've got your CAP on.



*Transition One*

# Make not fall down.

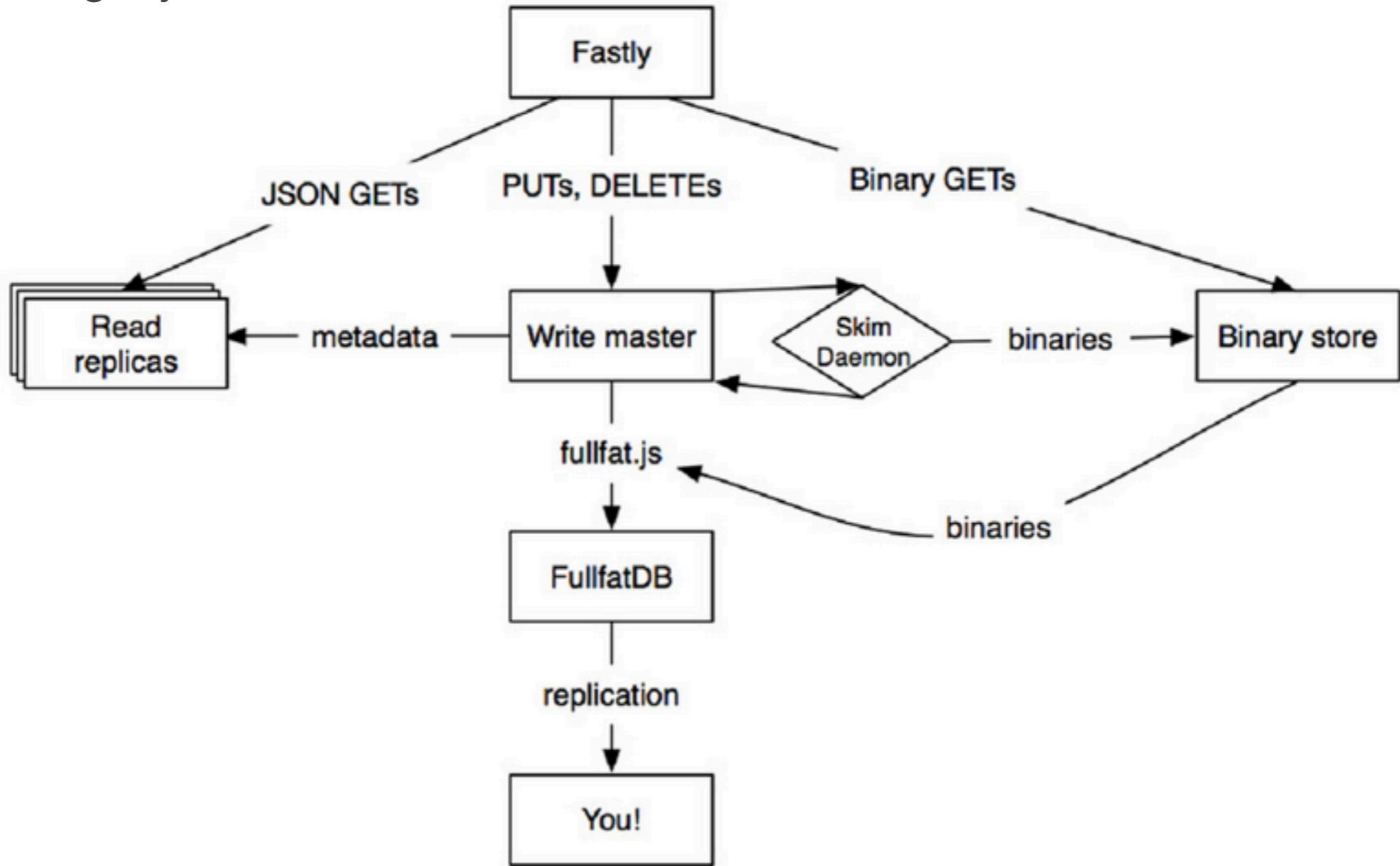
Make sure you've got your CAP on.



*Transition Two*

# Make not fall down.

Make sure you've got your CAP on.



*March 2014*

# This was good and bad for

Good news, then bad news, then more bad news

*Bad news!*

- More servers means more points of failure.
- Opted for CP (Consistency, Partition Tolerance) with “fast fail-over A” (availability)
- So ... **your builds are broken by design.**

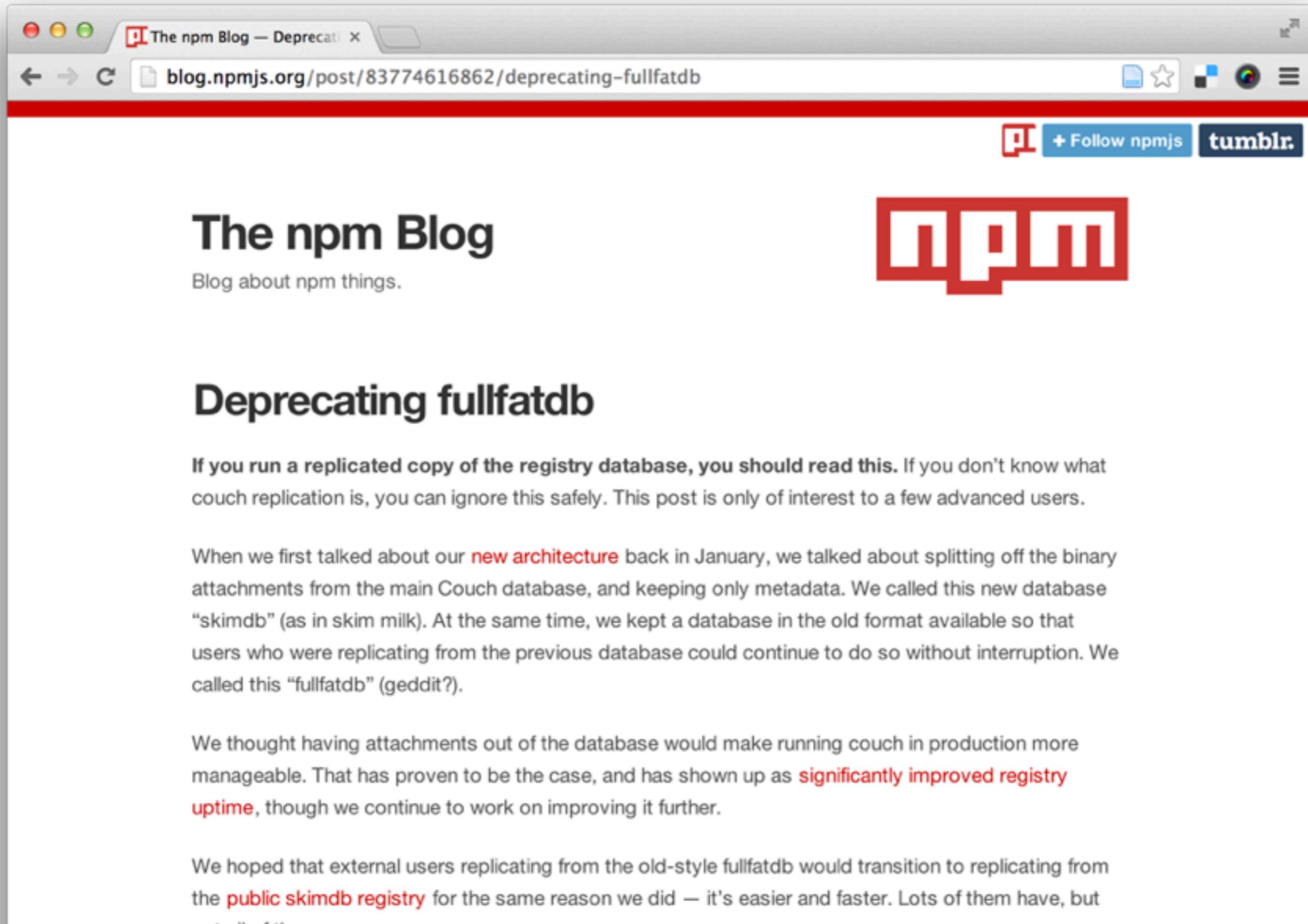
*Good news!*

- Made replicating JSON metadata awesome and easy.
- If you need AP with eventual C then

`npm c set registry https://registry.nodejitsu.com`

# Deprecating fullfatdb

Probably not the best idea ... *seriously*



The screenshot shows a web browser window with the following details:

- Address Bar:** The npm Blog — Deprecating fullfatdb (blog.npmjs.org/post/83774616862/deprecating-fullfatdb)
- Header:** The npm Blog (with a red 'p' logo), Follow npmjs, tumblr.
- Content Area:**
  - Section Header:** Deprecating fullfatdb
  - Text:** If you run a replicated copy of the registry database, you should read this. If you don't know what couch replication is, you can ignore this safely. This post is only of interest to a few advanced users.
  - Text:** When we first talked about our [new architecture](#) back in January, we talked about splitting off the binary attachments from the main Couch database, and keeping only metadata. We called this new database "skimdb" (as in skim milk). At the same time, we kept a database in the old format available so that users who were replicating from the previous database could continue to do so without interruption. We called this "fullfatdb" (geddit?).
  - Text:** We thought having attachments out of the database would make running couch in production more manageable. That has proven to be the case, and has shown up as [significantly improved registry uptime](#), though we continue to work on improving it further.
  - Text:** We hoped that external users replicating from the old-style fullfatdb would transition to replicating from the [public skimdb registry](#) for the same reason we did — it's easier and faster. Lots of them have, but

# Deprecating fullfatdb

Why is it a bad idea exactly?

- Breaks open replication through CouchDB
- All replication must be performed by npm-fullfat-registry
- But npm-fullfat-registry will **not be run in production by anyone.**



# I <3 CouchDB

And that's OK!

- People hate on CouchDB
- Because people don't understand it
- CouchDB is most of the time not the source of your problem
- Your data design is probably not taking advantage of how CouchDB works.
- *Man walks into a doctor and says “it hurts when I do this” .... get it?*



# npm has mutable documents

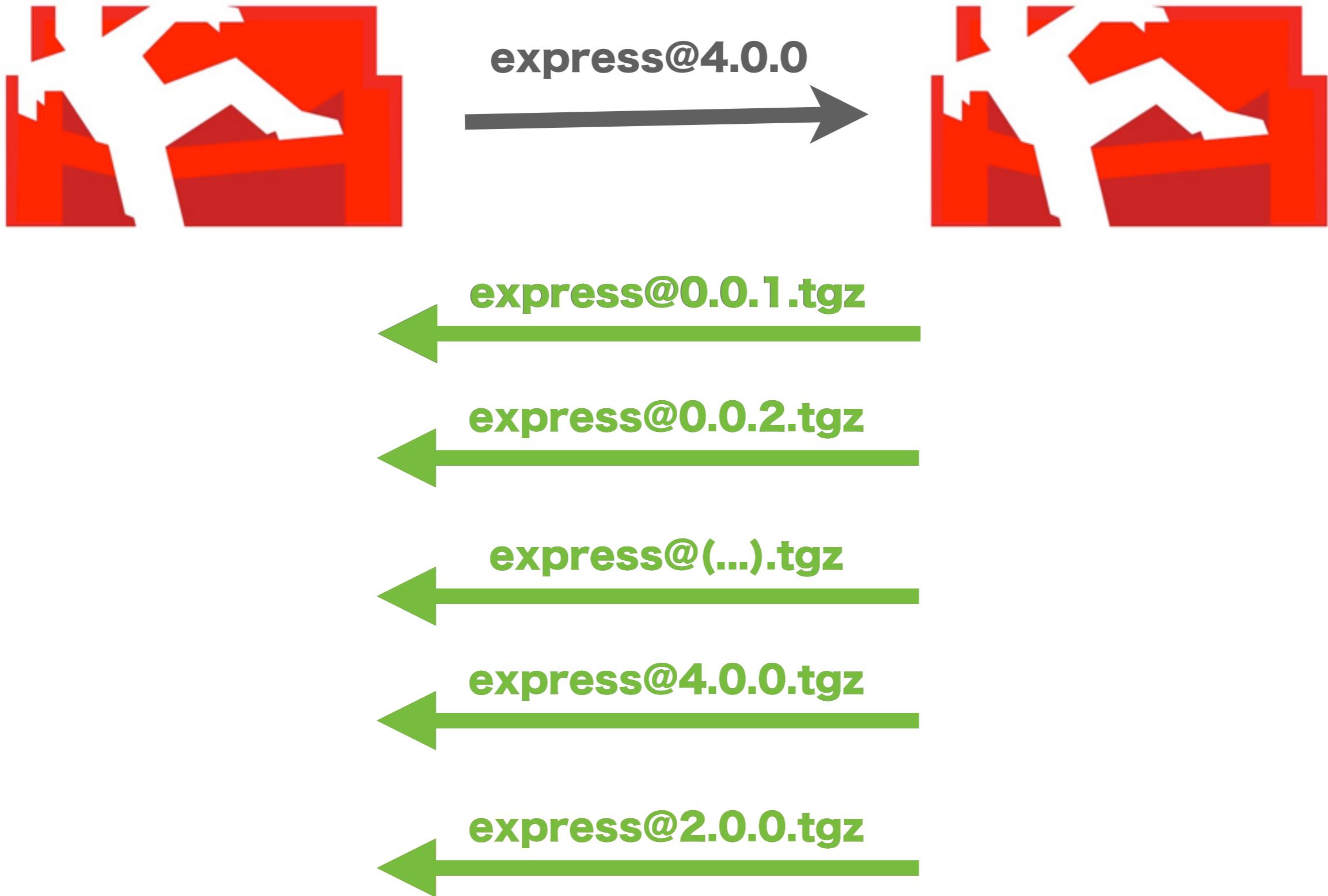
This makes replication bad and stuff. And stuff.



- Replication target gets the new JSON document.
- Replication target fetches attachment for express@4.0.0
- *Right?*
- **WRONG!**

# npm has mutable documents

What really happens on every npm publish



# npm with immutable documents

We can have our Couch and our attachments.

```
{  
  _id: "winston",  
  _rev: "182-f385b244a0f207487d56cc04ff836bec",  
  name: "winston",  
  description: "A multi-transport async logging library for Node.js",  
  - dist-tags: {  
    latest: "0.7.3"  
  },  
  - versions: {  
    + 0.2.11: {...},  
    + 0.3.3: {...},  
    + 0.3.4: {...},  
    + 0.3.5: {...},  
    + 0.4.0: {...},  
    + 0.4.1: {...},  
    - 0.5.0: {  
      name: "winston",  
      description: "A multi-transport async logging library for Node.js",  
      version: "0.5.0",  
      - author: {  
        name: "Charlie Robbins",  
        email: "charlie.robbins@gmail.com"  
      },  
      - contributors: [  
        - {  
          name: "Matthew Bergman",  
          email: "mzbphoto@gmail.com"  
        },  
        - {  
          name: "Marak Squires",  
          email: "marak.squires@gmail.com"  
        }  
      ]  
    }  
  }  
}
```

# npm with immutable documents

One attachment per document. One version per document.



- Replication target gets the new JSON document.
- Replication target fetches attachment for express@4.0.0
- *Right?*
- **YES YES YES YES!**



# **npm** with immutable documents

Coming soon to a CouchDB near you!

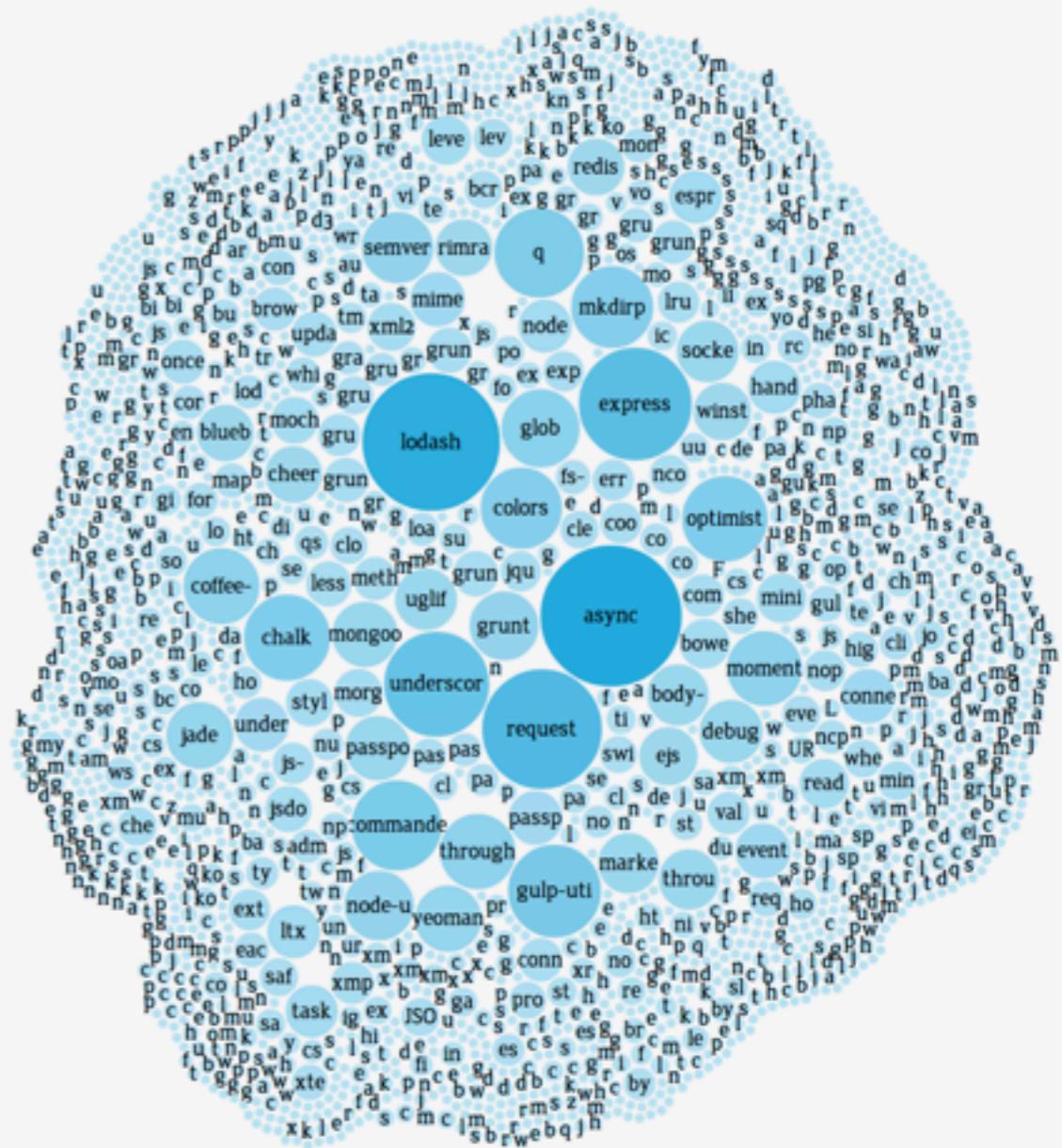
[github.com/nodejitsu/npm-ev-source](https://github.com/nodejitsu/npm-ev-source)

@jcrugzz

# Dependency graphs

## The $G = (V, E)$ kind of graph

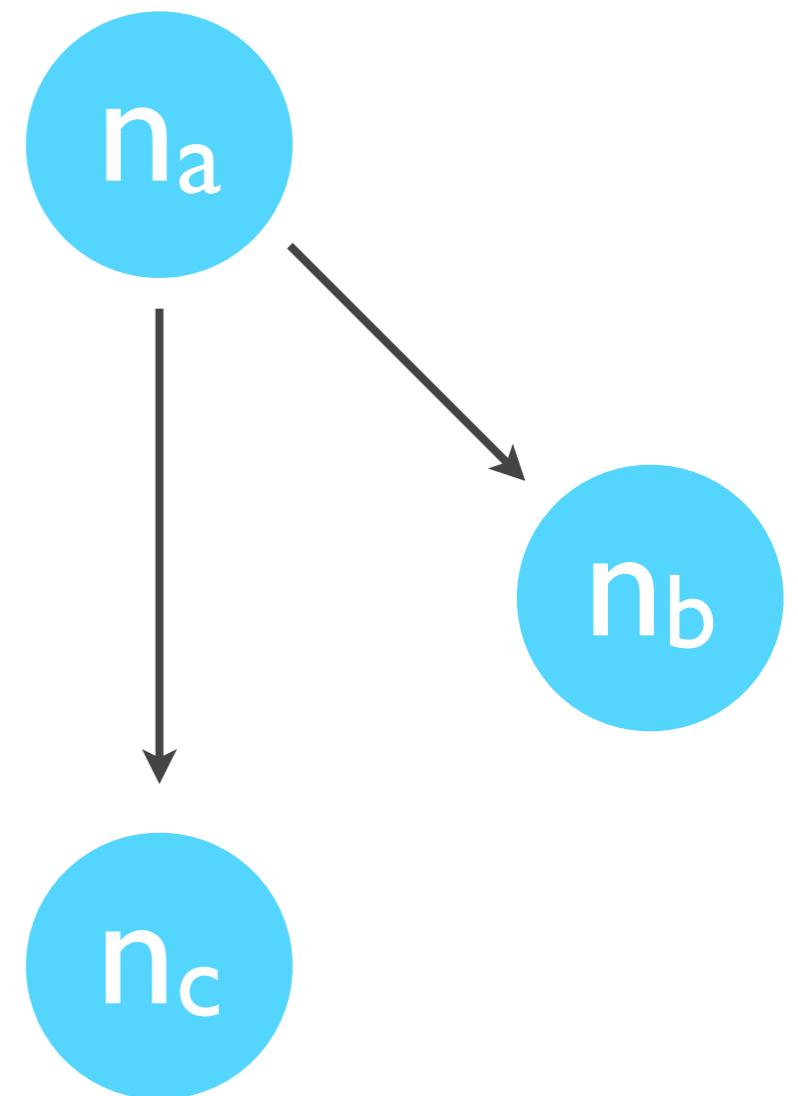
- Basic graph representation is not extremely helpful for visualizing package relationships.
  - But it does provide a basic structure for a graph search problem.



# Dependency graphs

The  $G = (V, E)$  kind of graph

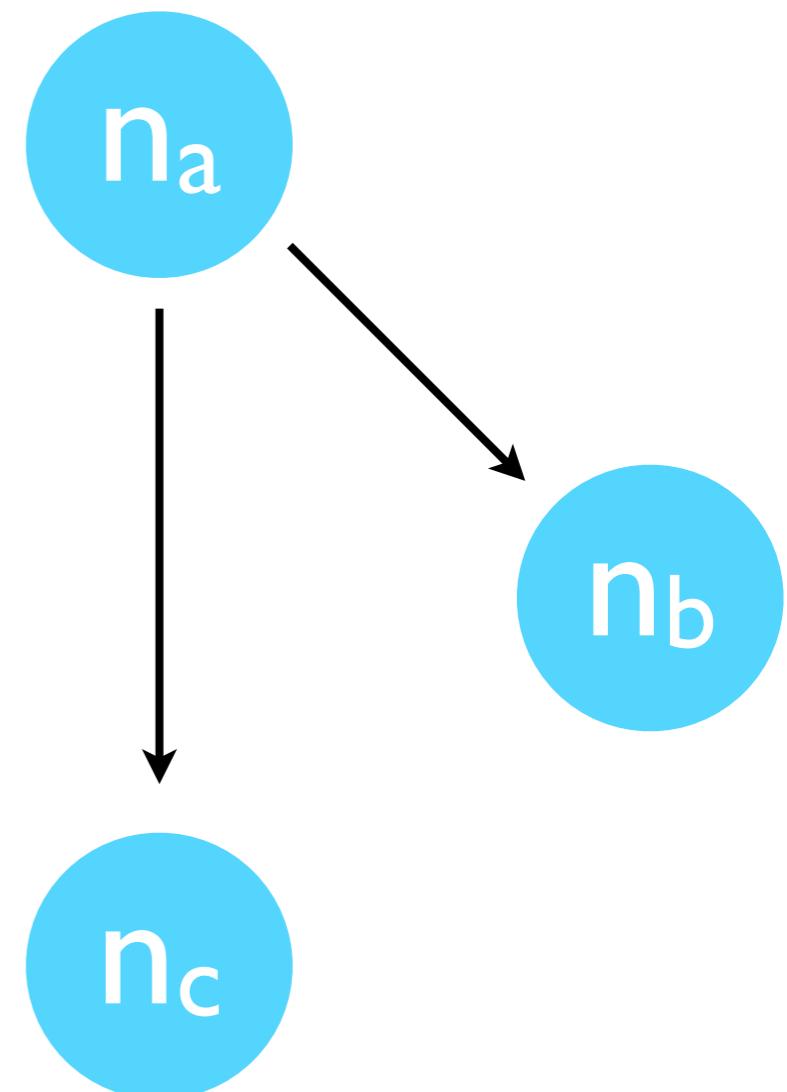
- To build our graph,  $G$ , we add a node (or vertex) for every package.
- We then add a colored edge from any node  $n_A$  to  $n_B$  if package A depends on package B.
- Edges are colored by dependency type: dependencies or devDependencies



# Dependency graphs

The  $G = (V, E)$  kind of graph

```
{  
  "name": "package-a",  
  "dependencies": {  
    "package-b": "~1.0.4",  
    "package-c": "~2.1.3"  
  },  
  "main": "./index.js"  
}
```



- Now imagine this graph for 70,000+ modules.

A massive, bright orange and yellow explosion dominates the center of the image, set against a backdrop of a clear blue sky and a green, hilly landscape. The explosion is highly detailed, with intense fire and smoke billowing upwards and outwards. The text 'MIND. EXPLOSION.' is overlaid in the center of the explosion.

**MIND. EXPLOSION.**

Actually.  
It's not so bad.

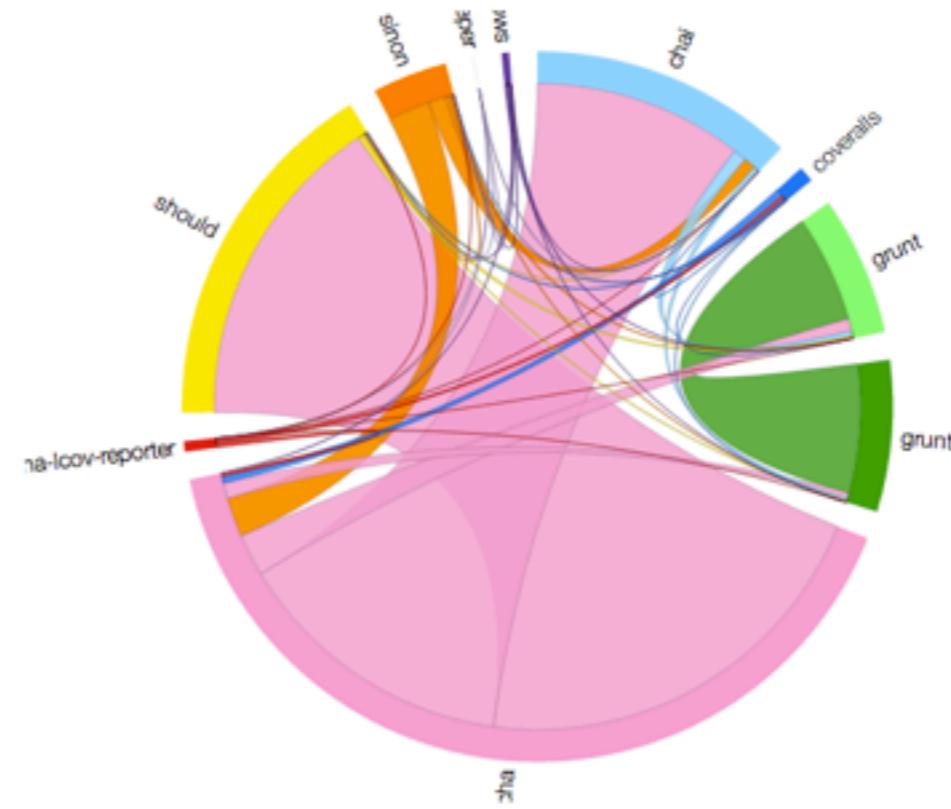
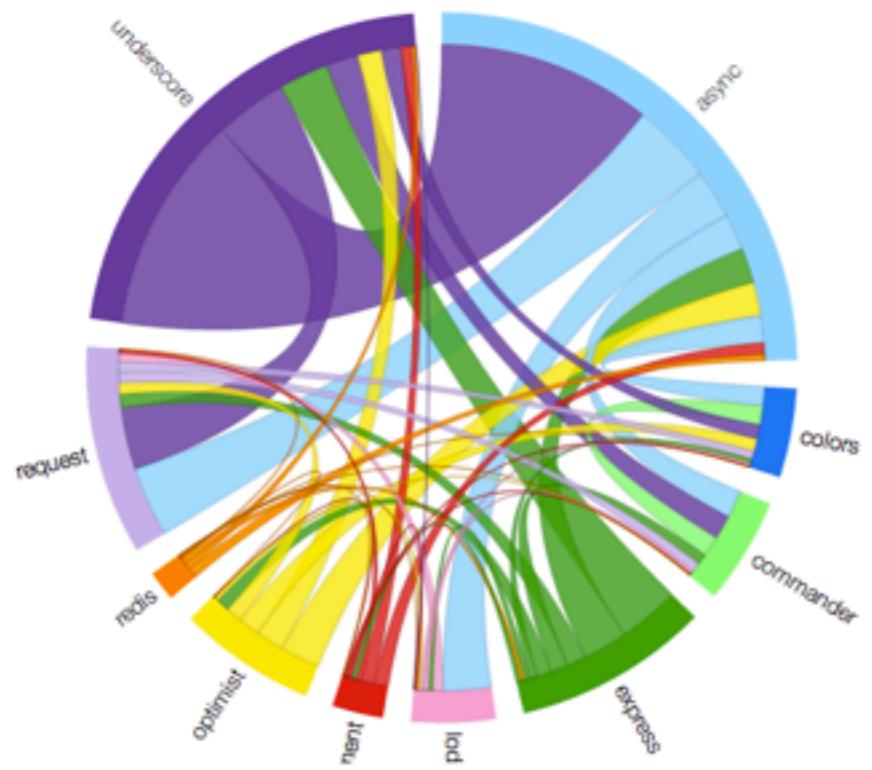


# Dependency graphs

Ok, so what is this useful for?

- There are tons of useful applications of dependency graphs.
- Lets consider two.

## First: Codependencies



# Codependencies?

Well, technically co(\*)dependencies.

- Codependencies answer the question “*people who depend on package A also depend on ...*”

[ "package", "codep", "thru" ]



<3 CouchDB!

# Codependencies?

Thanks CouchDB!

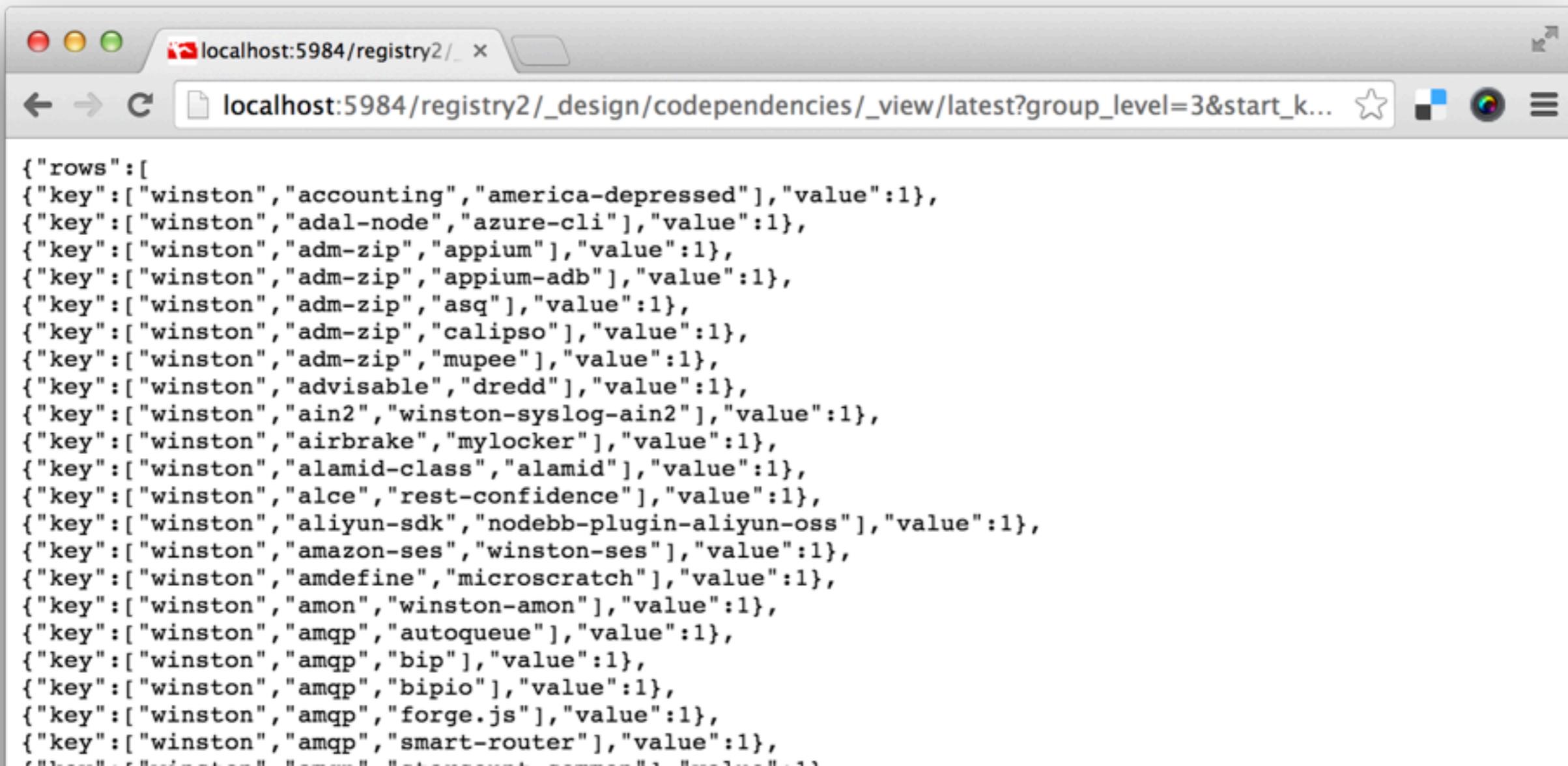
- We can get all of this from a **simple CouchDB view**.

```
for (var dep in d) {  
  dep = dep.trim();  
  for (var codep in d) {  
    codep = codep.trim();  
    if (dep !== codep) {  
      emit([dep, codep, doc._id], 1)  
    }  
  }  
}
```

# Codependencies?

Thanks CouchDB!

```
group_level=3
&start_key=["winston"]
&end_key=["winston",{}]
```



```
{"rows": [
  {"key": ["winston", "accounting", "america-depressed"], "value": 1},
  {"key": ["winston", "adal-node", "azure-cli"], "value": 1},
  {"key": ["winston", "adm-zip", "appium"], "value": 1},
  {"key": ["winston", "adm-zip", "appium-adb"], "value": 1},
  {"key": ["winston", "adm-zip", "asq"], "value": 1},
  {"key": ["winston", "adm-zip", "calipso"], "value": 1},
  {"key": ["winston", "adm-zip", "mupee"], "value": 1},
  {"key": ["winston", "advisable", "dredd"], "value": 1},
  {"key": ["winston", "ain2", "winston-syslog-ain2"], "value": 1},
  {"key": ["winston", "airbrake", "mylocker"], "value": 1},
  {"key": ["winston", "alamid-class", "alamid"], "value": 1},
  {"key": ["winston", "alce", "rest-confidence"], "value": 1},
  {"key": ["winston", "aliyun-sdk", "nodebb-plugin-aliyun-oss"], "value": 1},
  {"key": ["winston", "amazon-ses", "winston-ses"], "value": 1},
  {"key": ["winston", "amdefine", "microscratch"], "value": 1},
  {"key": ["winston", "amon", "winston-amon"], "value": 1},
  {"key": ["winston", "amqp", "autoqueue"], "value": 1},
  {"key": ["winston", "amqp", "bip"], "value": 1},
  {"key": ["winston", "amqp", "bipio"], "value": 1},
  {"key": ["winston", "amqp", "forge.js"], "value": 1},
  {"key": ["winston", "amqp", "smart-router"], "value": 1},
  {"key": ["winston", "amqp", "stompjs"], "value": 1}
]}
```

# Codependencies

---

The meat of the analysis

- So this is all well and good, but what the heck are you doing?!?!

For module **NAME** generate a matrix by:

- **Rank** codependencies based on number of times they appear
- For each codependency **C** in the **SET** of the top **N**:  
**Rank** **SET** – **{C}** by number of times they appear to create **Row[C]**

# Codependencies

Understanding codependencies through winston

- This last step yields a matrix for the codependency relationship:

	asyn...	expr...	opti...	requ...	unde...
asyn...	<b>0.0000</b>	<b>553.0000</b>	<b>534.0000</b>	<b>837.0000</b>	<b>1359.0000</b>
expr...	<b>553.0000</b>	<b>0.0000</b>	<b>314.0000</b>	<b>365.0000</b>	<b>648.0000</b>
opti...	<b>534.0000</b>	<b>314.0000</b>	<b>0.0000</b>	<b>335.0000</b>	<b>448.0000</b>
requ...	<b>837.0000</b>	<b>365.0000</b>	<b>335.0000</b>	<b>0.0000</b>	<b>786.0000</b>
unde...	<b>1359.0000</b>	<b>648.0000</b>	<b>448.0000</b>	<b>786.0000</b>	<b>0.0000</b>

# Codependencies

Understanding codependencies through winston

- Now we need to weight the matrix based on the overall appearance of these codependencies

240		async		0.2761
207		underscore		0.2382
163		express		0.1875
133		request		0.1530
126		optimist		0.1449
869	<b>total</b>			

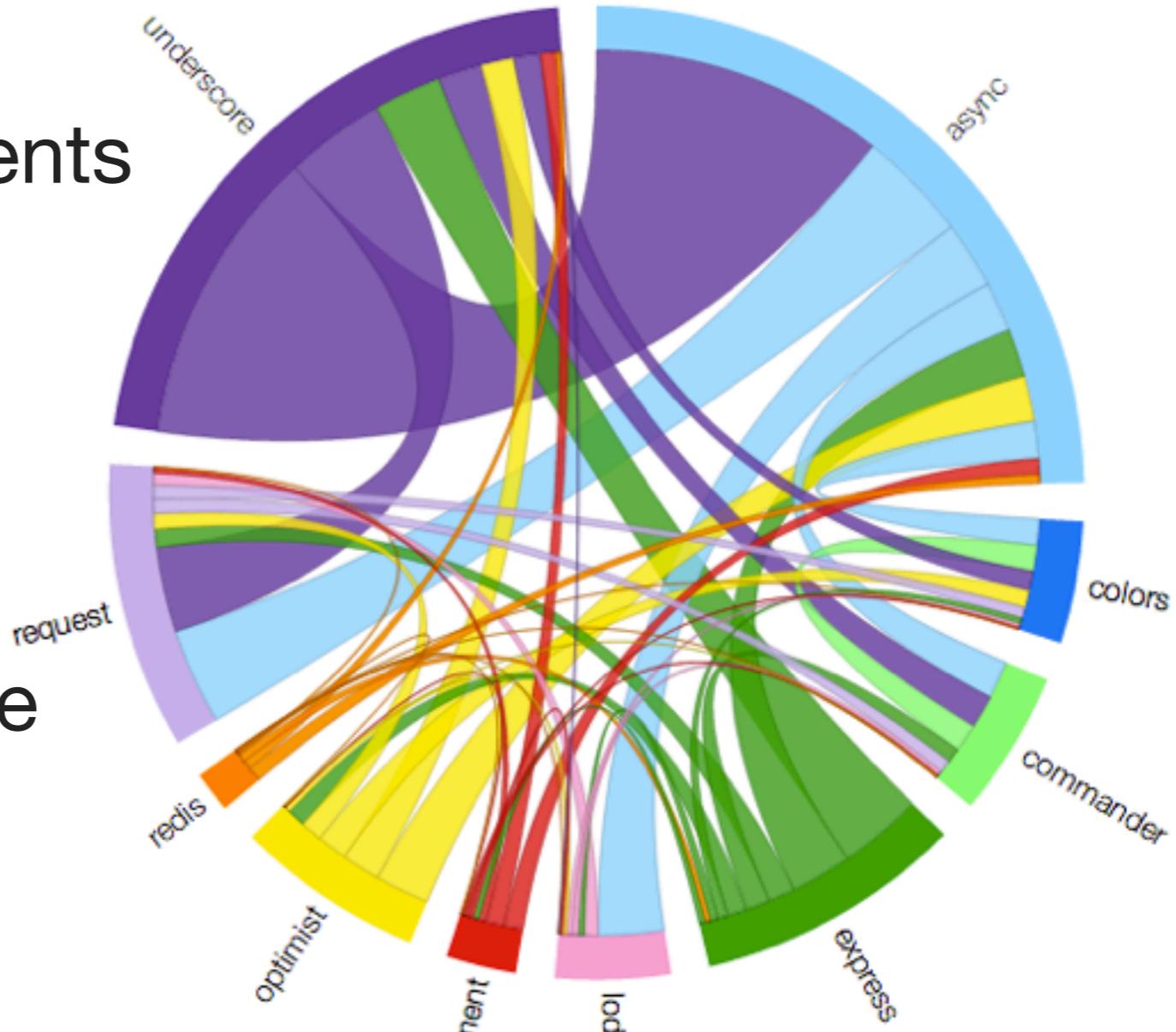
# Codependencies

Reading the tea leaves of dense data visualization

- **The size of the arc** represents the degree of the codependency relationship with the parent module.

- **The size of the chord** represents the degree of the codependency relationship between each pair.

- **The color of the chord** represents the “dominant” module between the pair.





PAUSE FOR  
DEMO

COMEDY C

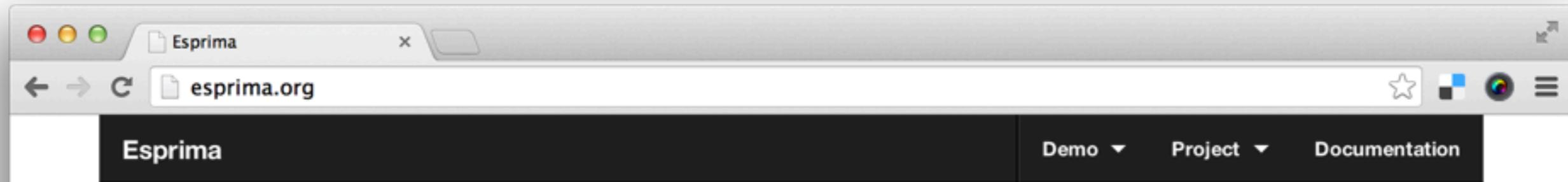
# I should use static analysis with **npm**

ANALYZE ALL THE THINGS!



# I should use static analysis with

How the what exactly?



**ECMAScript** parsing infrastructure for multipurpose analysis

Esprima is a high performance, standard-compliant **ECMAScript** parser written in **ECMAScript** (also popularly known as **JavaScript**).

## Features

- Full support for ECMAScript 5.1 ([ECMA-262](#))
- Sensible [syntax tree format](#), compatible with Mozilla [Parser AST](#)
- Optional tracking of syntax node location (index-based and line-column)
- Heavily tested (> 700 [tests](#) with [full code coverage](#))
- [Partial support](#) for ECMAScript 6

Esprima serves as an important **building block** for some JavaScript language tools, from [code instrumentation](#) to [editor autocomplete](#).

```
1 var capitalDb = {  
2   Indonesia: 'Jakarta',  
3   Germany: 'Berlin',  
4   Norway: 'Oslo'  
5 };  
6  
7 // Property completion: "capitalDb." and press Ctrl+Space.  
8 capitalDb.  
9           Germany : String  
          Indonesia : String
```

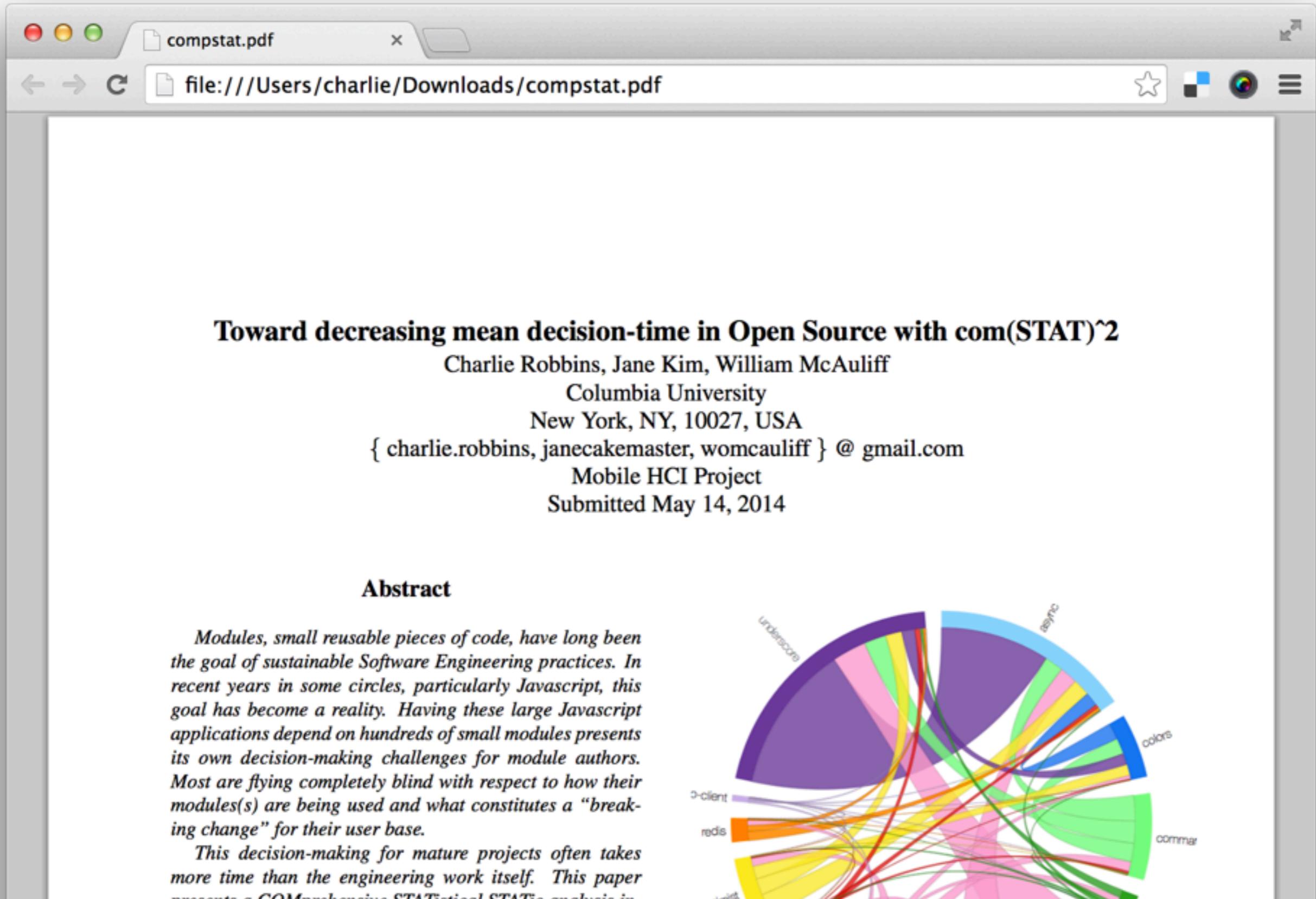
Once the full syntax tree is obtained, various **static code analysis** can be applied to give an insight to the code: [syntax visualization](#), [code validation](#), [editing autocomplete](#) (with type inferencing) and [many others](#).

Regenerating the code from the syntax tree permits a few different types of **code transformation**, from a simple [rewriting](#) (with specific formatting) to a more complicated [minification](#).

Esprima runs on many popular web browsers, as well as other ECMAScript platforms such as [Rhino](#), [Nashorn](#), and

# I should use static analysis with

Why the what precisely?



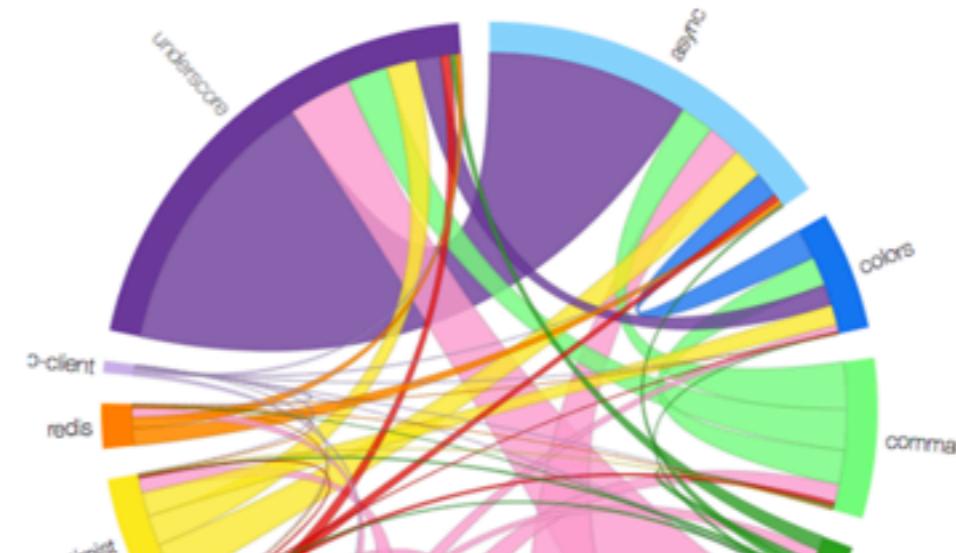
**Toward decreasing mean decision-time in Open Source with com(STAT)<sup>2</sup>**

Charlie Robbins, Jane Kim, William McAuliffe  
Columbia University  
New York, NY, 10027, USA  
{ charlie.robbins, janecakemaster, womcauliff } @ gmail.com  
Mobile HCI Project  
Submitted May 14, 2014

**Abstract**

*Modules, small reusable pieces of code, have long been the goal of sustainable Software Engineering practices. In recent years in some circles, particularly Javascript, this goal has become a reality. Having these large Javascript applications depend on hundreds of small modules presents its own decision-making challenges for module authors. Most are flying completely blind with respect to how their module(s) are being used and what constitutes a “breaking change” for their user base.*

*This decision-making for mature projects often takes more time than the engineering work itself. This paper presents a COMprehensive STATistical STATIC analysis in*



# I should use static analysis with

---

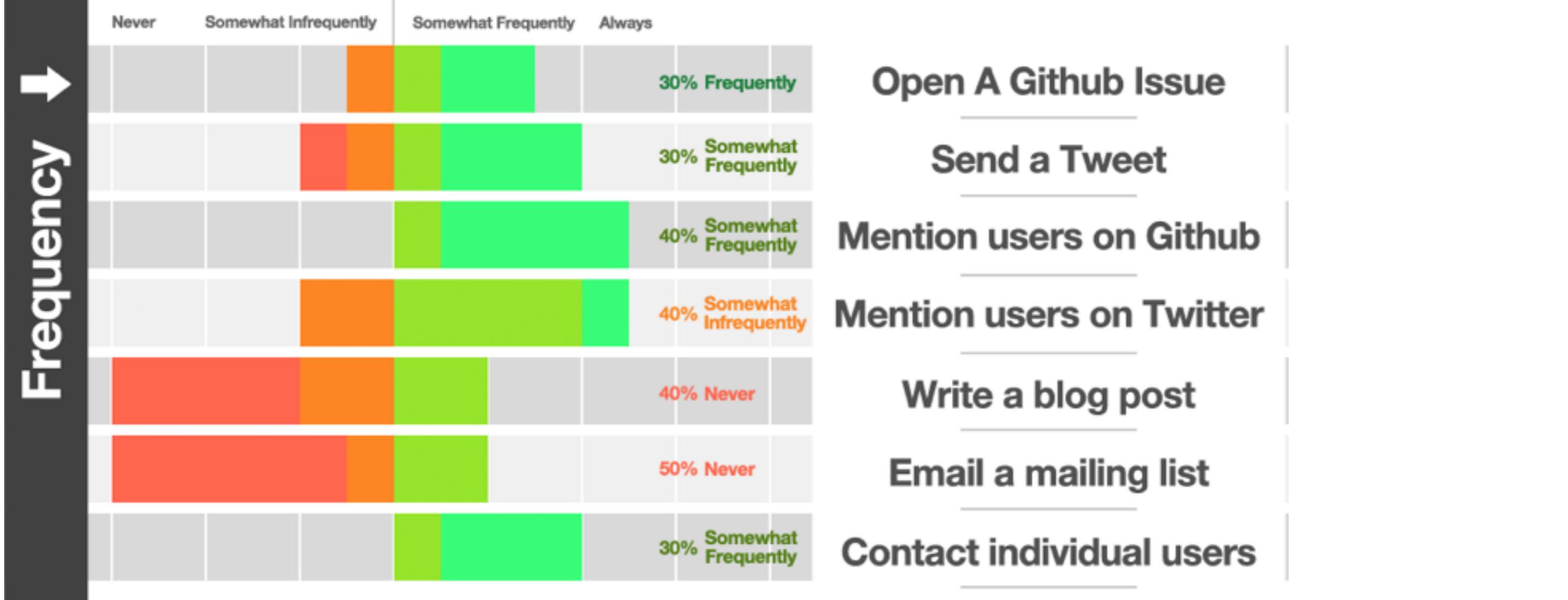
Why the what precisely?

**SMALL GROUP  
OF 8 EXPERTS**

I should use static analysis with 

Why the what precisely?

**MOST EFFECTIVE**  
**METHODS USED**  
**MOST INFREQUENTLY**



## Open A Github Issue

### Send a Tweet

### Mention users on Github

### Mention users on Twitter

### Write a blog post

### Email a mailing list

### Contact individual users

## Open A Github Issue

### Send a Tweet

### Mention users on Github

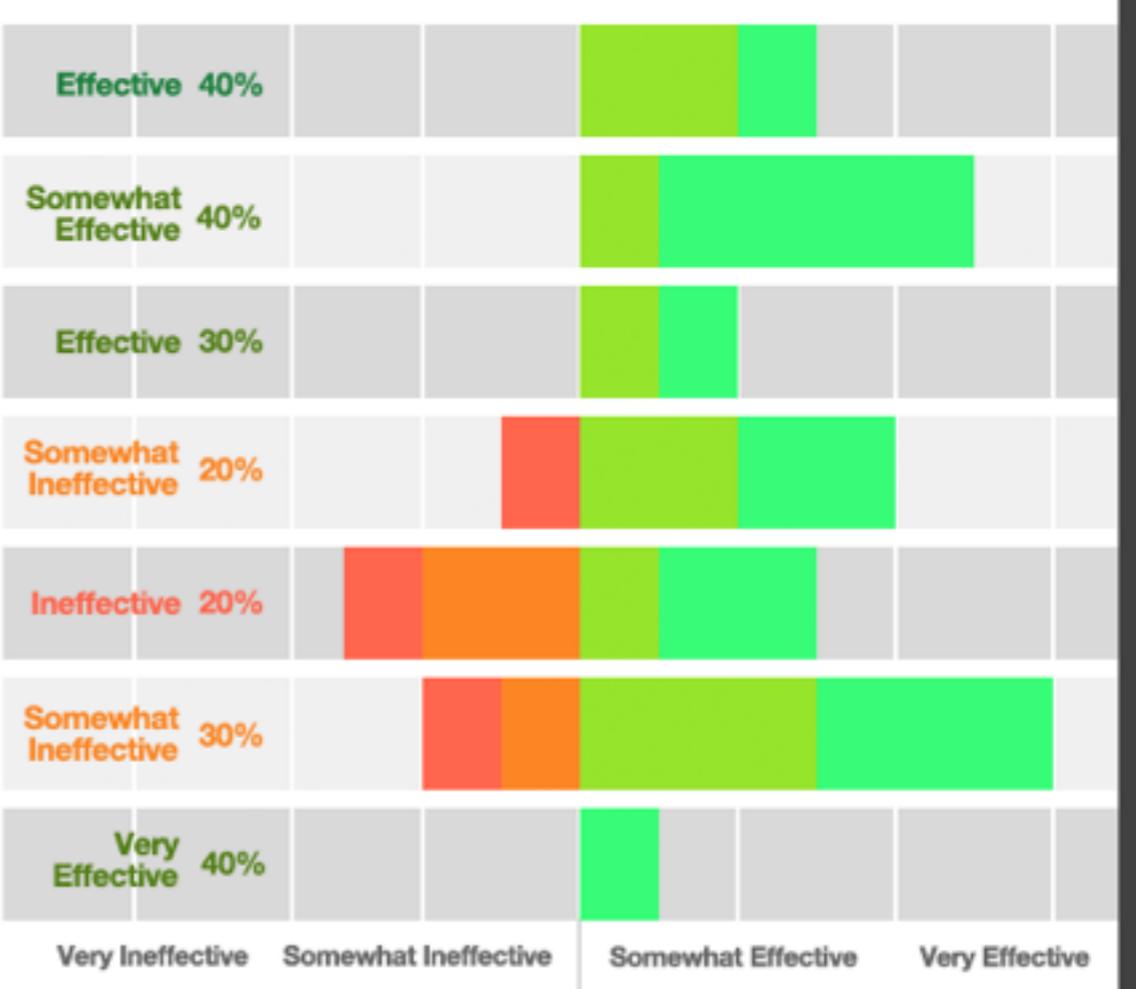
### Mention users on Twitter

### Write a blog post

### Email a mailing list

### Contact individual users

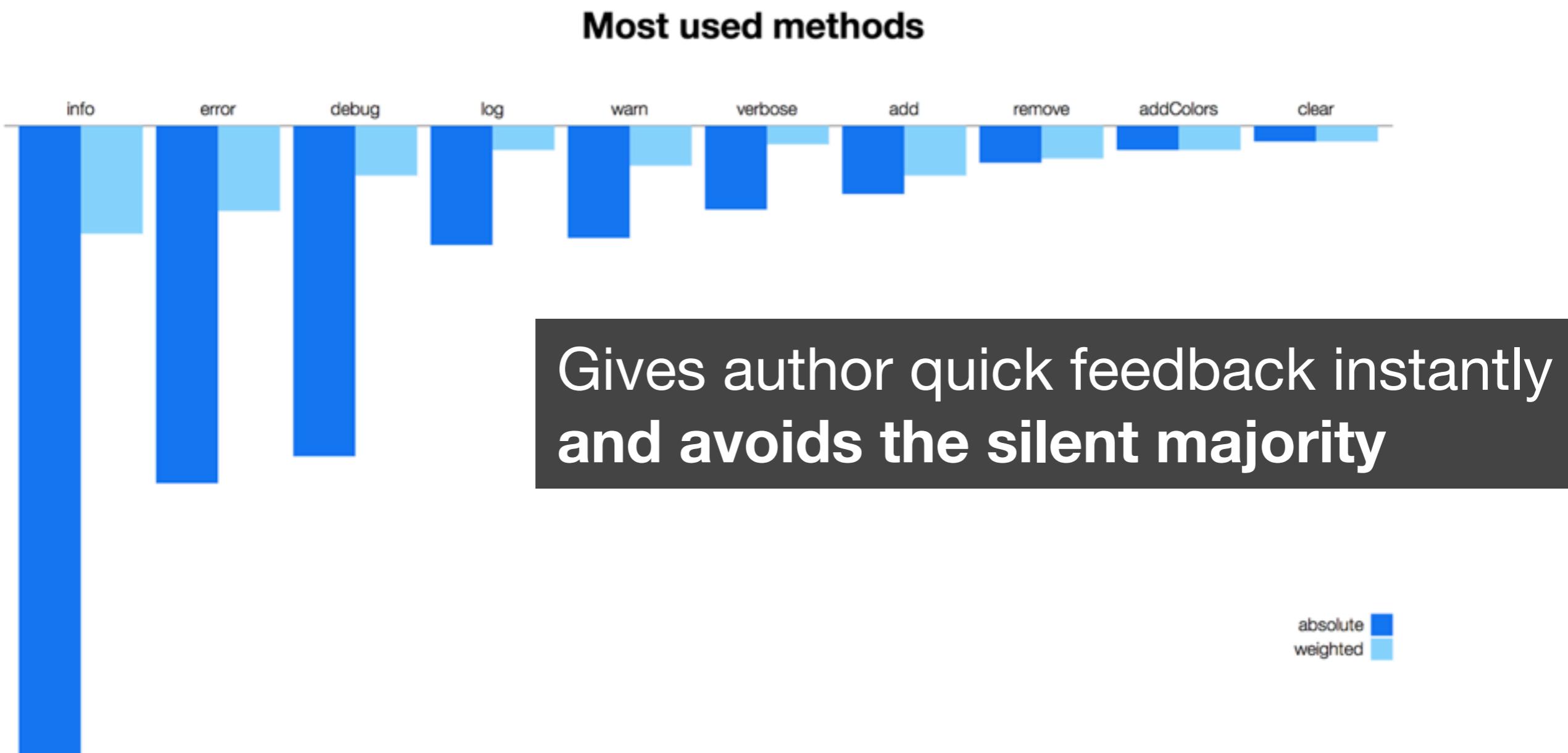
↑ Effectiveness

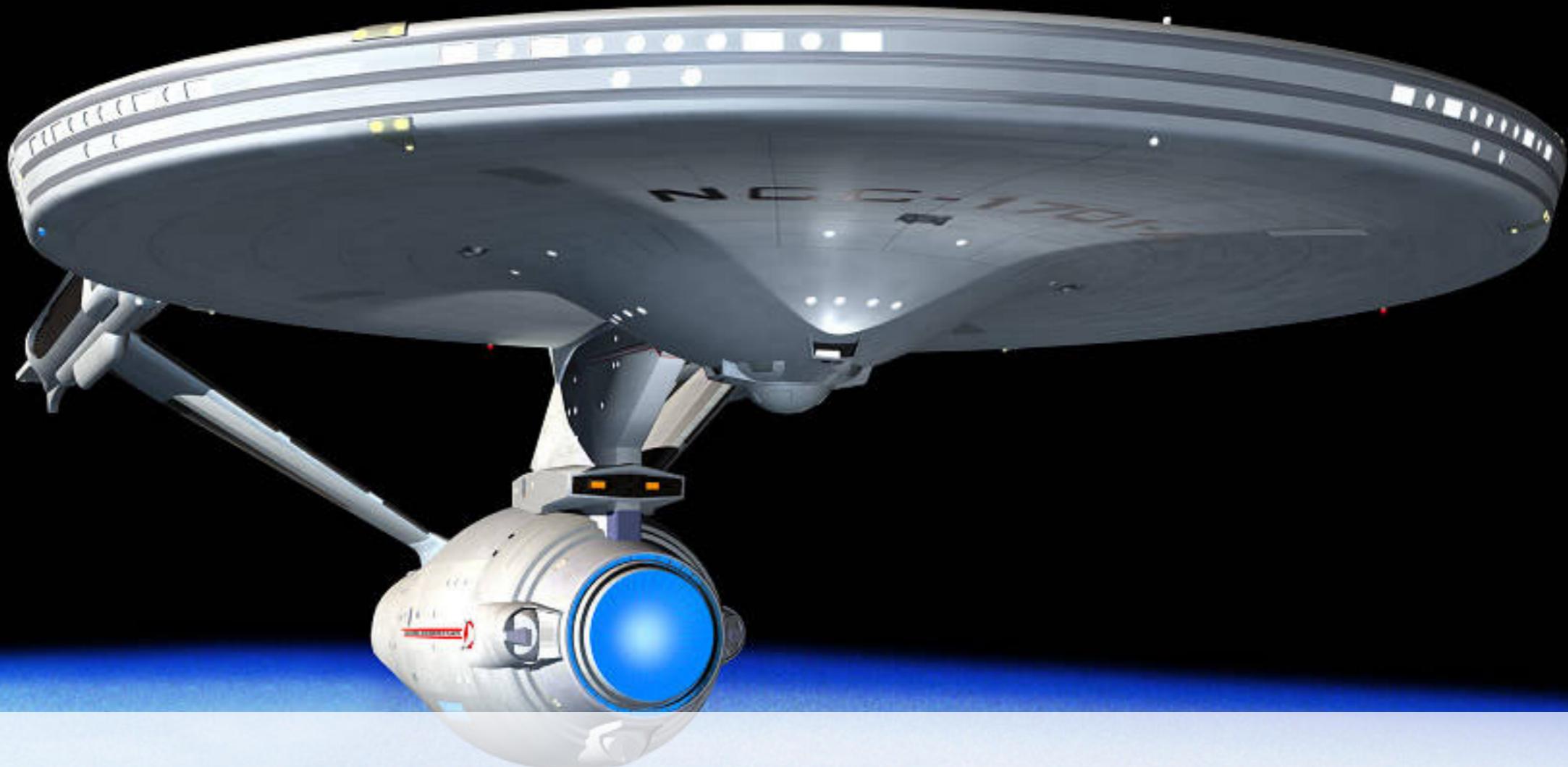


# I should use static analysis with

GET TO THE POINT ALREADY CHARLIE!

Using **esprima** and a **dependency graph** we were able to perform rudimentary analysis of hottest code paths.





**PAUSE FOR  
OPEN SOURCE**

# Using private modules with

Problem. Solved. Headshot.

# Using private modules with

Well. Sort of. There might be this “scoped module” thing.

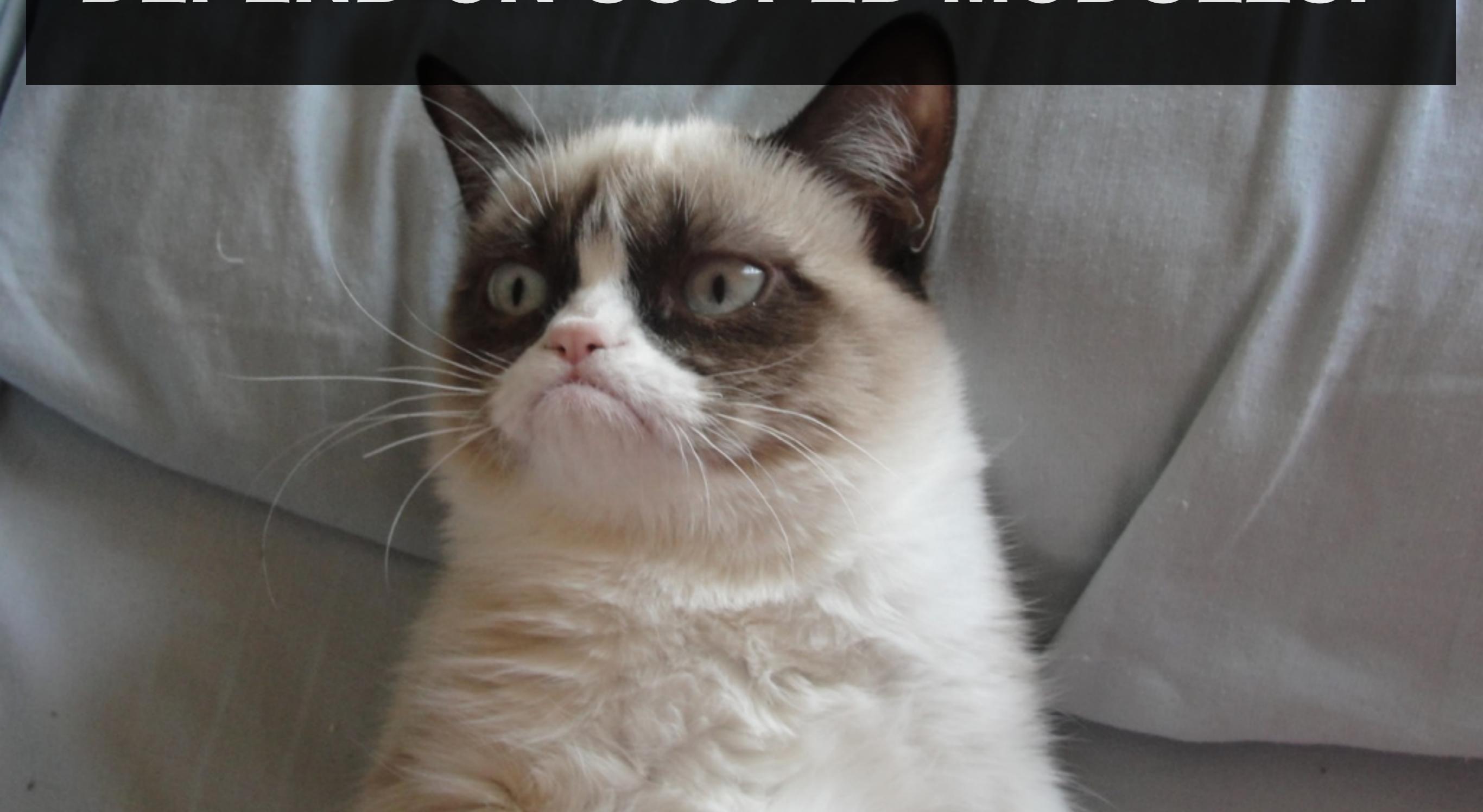
```
{  
  "name": "package-a",  
  "dependencies": {  
    "package-b": "~1.0.4",  
    "package-c": "~2.1.3"  
  },  
  "main": "./index.js"  
}
```

Remember  
this?

Now what  
about this?

```
{  
  "name": "package-a",  
  "dependencies": {  
    "@org/ppkg-b": "~1.0.4",  
    "package-c": "~2.1.3"  
  },  
  "main": "./index.js"  
}
```

**GLOBAL MODULES CANNOT  
DEPEND ON SCOPED MODULES!**





# **WARNING: shameless plug**

for  **nodejitsu**

---

I will be brief. I promise.

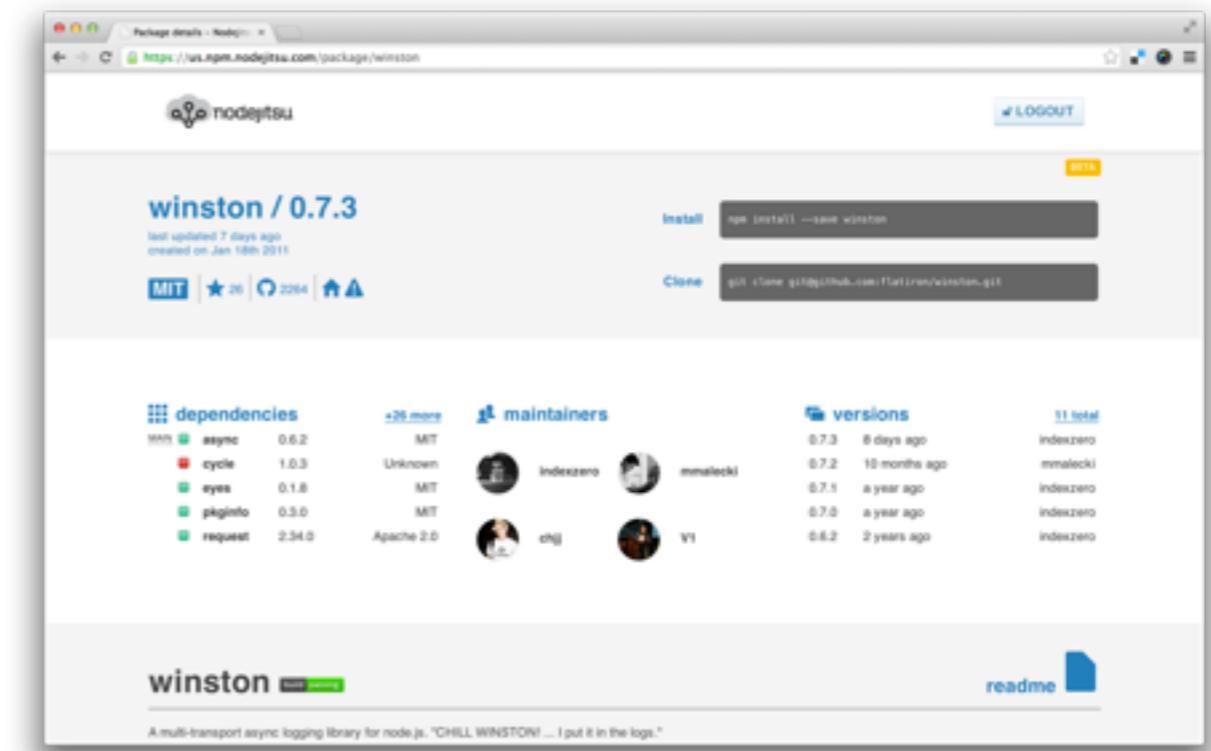
# What problem do we solve for **npm**?



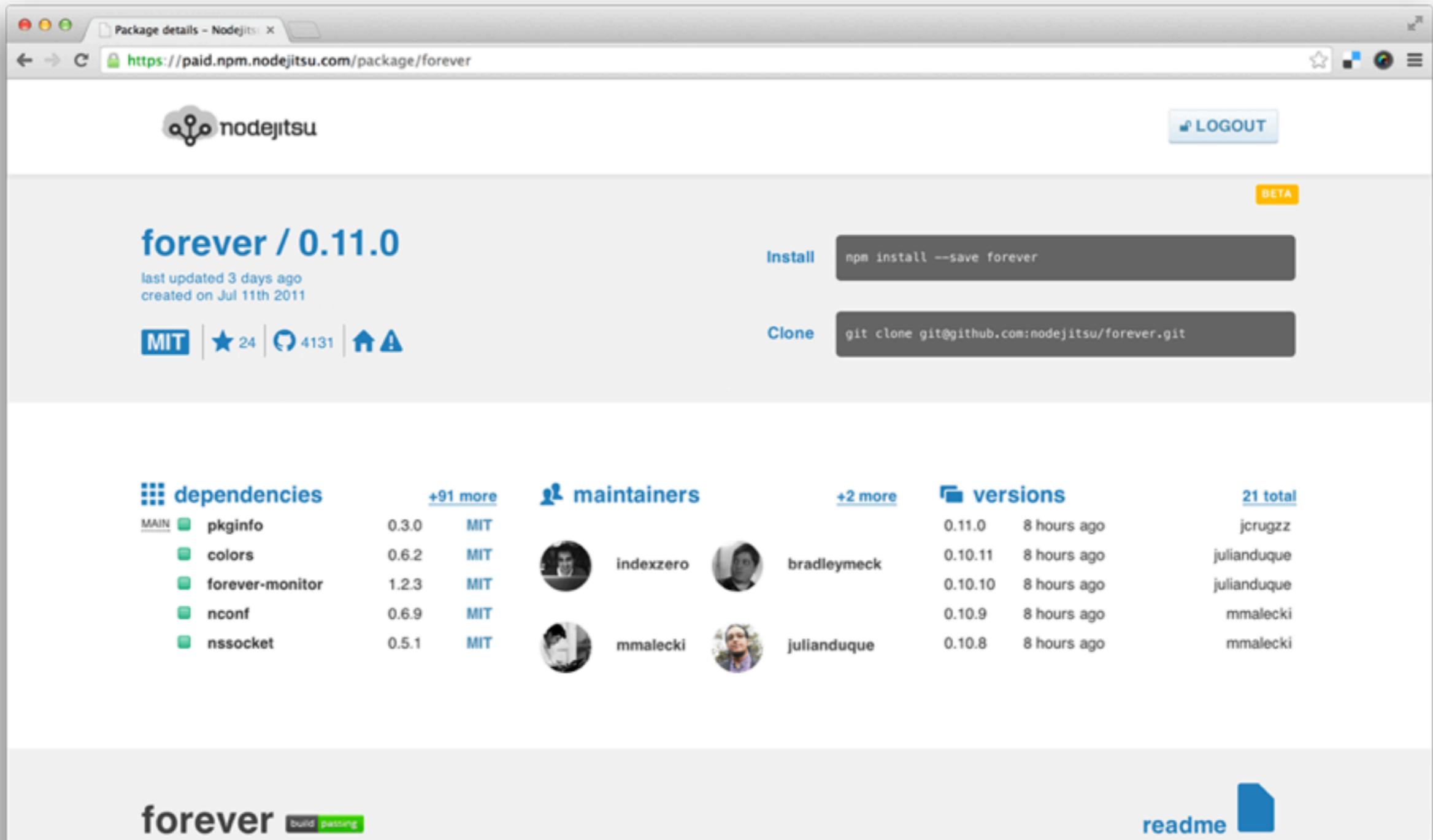
**Developers** have **questions** about their **applications**

We increase value & avoid wasted time by answering these questions

- *Where do I store my private code?*
- *Is my Node.js application **secure**? Against what vulnerabilities?*
- *How do I **deploy** my application?*
- *Am I at **legal risk** because of Open Source licenses?*
- *What module should I use to do <feature> in my app?*
- *Is this Open Source module **good**?*



# Github for production Javascript



Package details - Nodejitsu

<https://paid.npm.nodejitsu.com/package/forever>

 nodejitsu LOGOUT BETA

## forever / 0.11.0

last updated 3 days ago  
created on Jul 11th 2011

**Install** `npm install --save forever`

**Clone** `git clone git@github.com:nodejitsu/forever.git`

### dependencies

MAIN	pkinfo	0.3.0	MIT
	colors	0.6.2	MIT
	forever-monitor	1.2.3	MIT
	nconf	0.6.9	MIT
	nssocket	0.5.1	MIT

[+91 more](#)

### maintainers

indexzero	bradleymeck
	
	

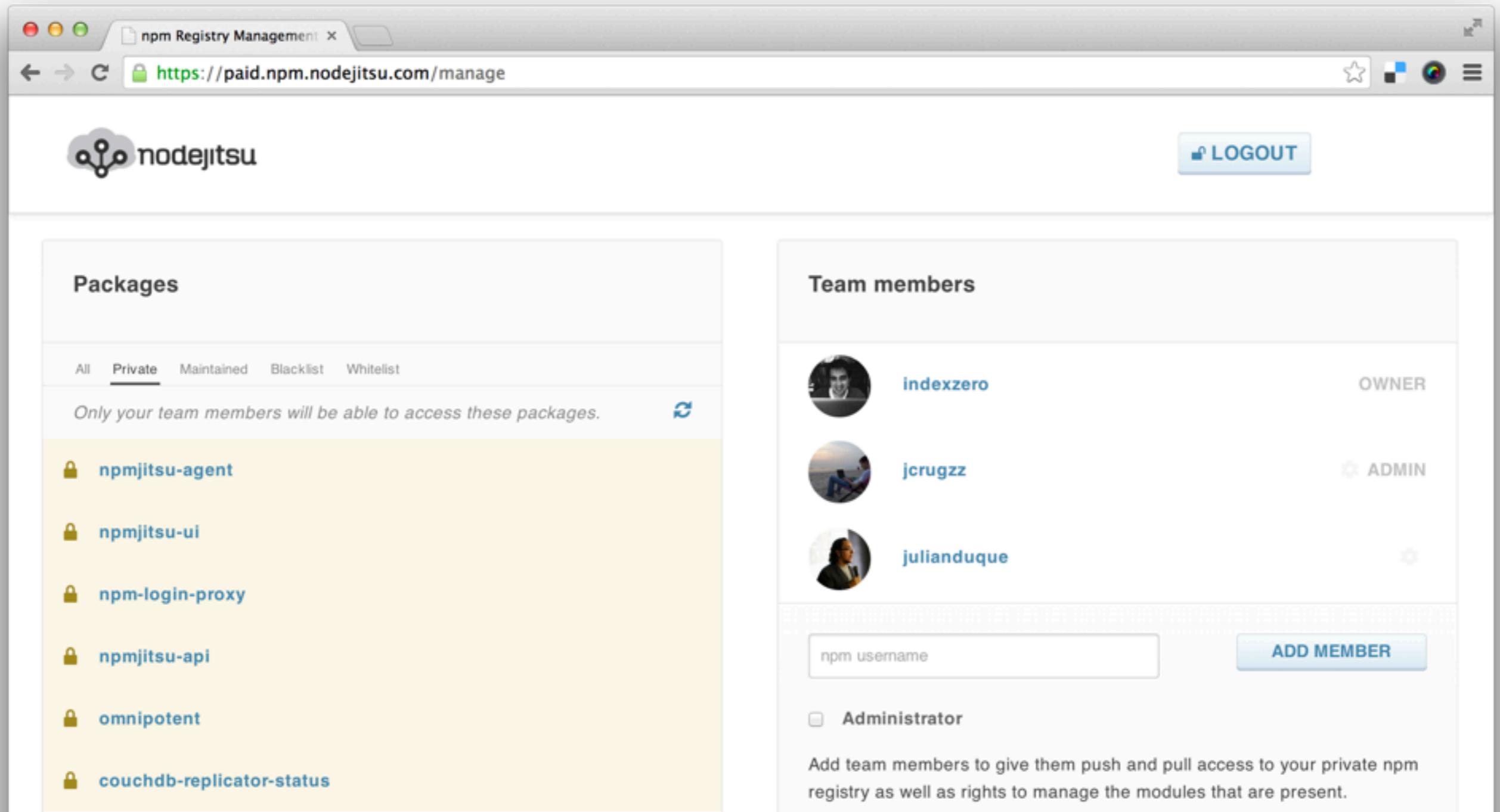
[+2 more](#)

### versions

21 total
0.11.0 8 hours ago jcrugzz
0.10.11 8 hours ago julianduque
0.10.10 8 hours ago julianduque
0.10.9 8 hours ago mmalecki
0.10.8 8 hours ago mmalecki

**forever**  **readme** 

# Manage your team. Manage your code.



The screenshot shows a web browser window for 'npm Registry Management' at <https://paid.npm.nodejitsu.com/manage>. The interface is divided into two main sections: 'Packages' on the left and 'Team members' on the right.

**Packages Section:**

- Header: nodejitsu
- Logout button: LOGOUT
- Filter buttons: All, **Private**, Maintained, Blacklist, Whitelist
- Text: Only your team members will be able to access these packages.
- List of private packages:
  - npmjitsu-agent
  - npmjitsu-ui
  - npm-login-proxy
  - npmjitsu-api
  - omnipotent
  - couchdb-replicator-status

**Team members Section:**

- Header: Team members
- List of team members:

	indexzero	OWNER
	jcrugzz	ADMIN
	julianduque	
- Form fields:
  - npm username:
  - Administrator:
  - ADD MEMBER button
- Text: Add team members to give them push and pull access to your private npm registry as well as rights to manage the modules that are present.

# Try it today

# nodejitsu.com/signup

The screenshot shows the Nodejitsu service status page. At the top, there's a search bar labeled "Search packages". Below it is a "Registry status" section featuring a world map where most regions are green (indicating "C" or "CM" status), while some areas like Europe and Australia are orange (indicating "C" status). To the left of the map is a list of mirrors:

- npm Inc. registry.npmjs.com C
- npm Inc. (fullfatdb) fullfatdb.npmjs.com C
- Nodejitsu registry.nodejitsu.com C
- Nodejitsu (IrisCouch) isaacs.ircouch.com C
- StrongLoop Inc. npm.strongloop.com C
- European mirror registry.npmjs.eu CM
- Chinese mirror registry.cnpmjs.org CM
- Australian mirror registry.npmjs.au CM

On the right, there are two charts: a line graph for "RESPONSE TIME" showing values between 0 and 400 ms with a current value of 64 ms, and a heatmap for "REPLICATION LAG" showing values from 0 to 24 minutes with a current value of 0 minutes. The heatmap legend indicates time intervals: day\*, hour, minute, and none.

At the bottom, there are links for "nodejitsu" and "npm", and social media icons for Twitter and GitHub. The footer also includes links for "Nodejitsu.com", "Blog", and "Service Status".



**nodejitsu**

**FIN**