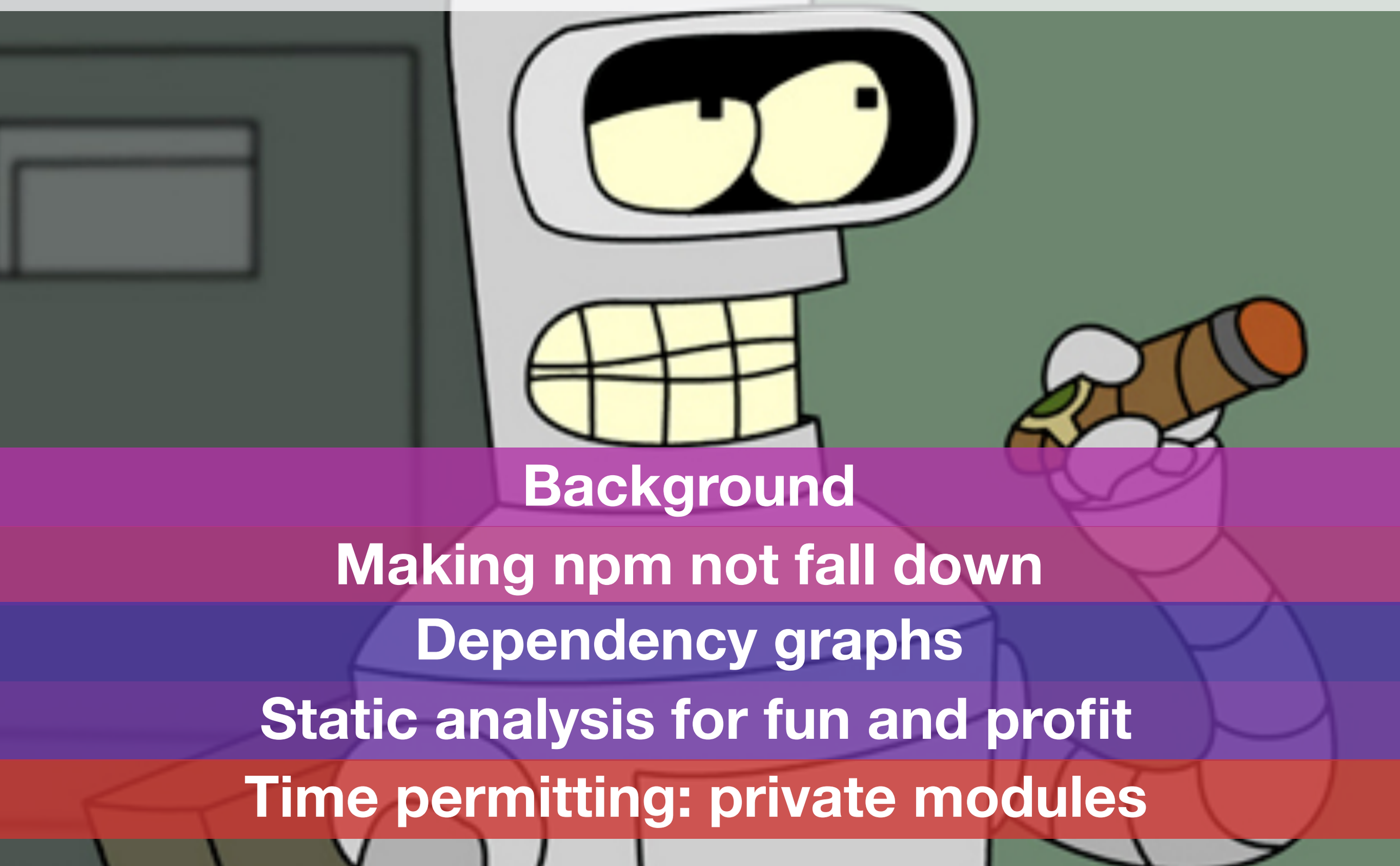
npm

THE FINAL FRONTIER

‘ello. What’s all this then?

And why should I pay attention?



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

Until January 2013, IrisCouch ran the npm registry

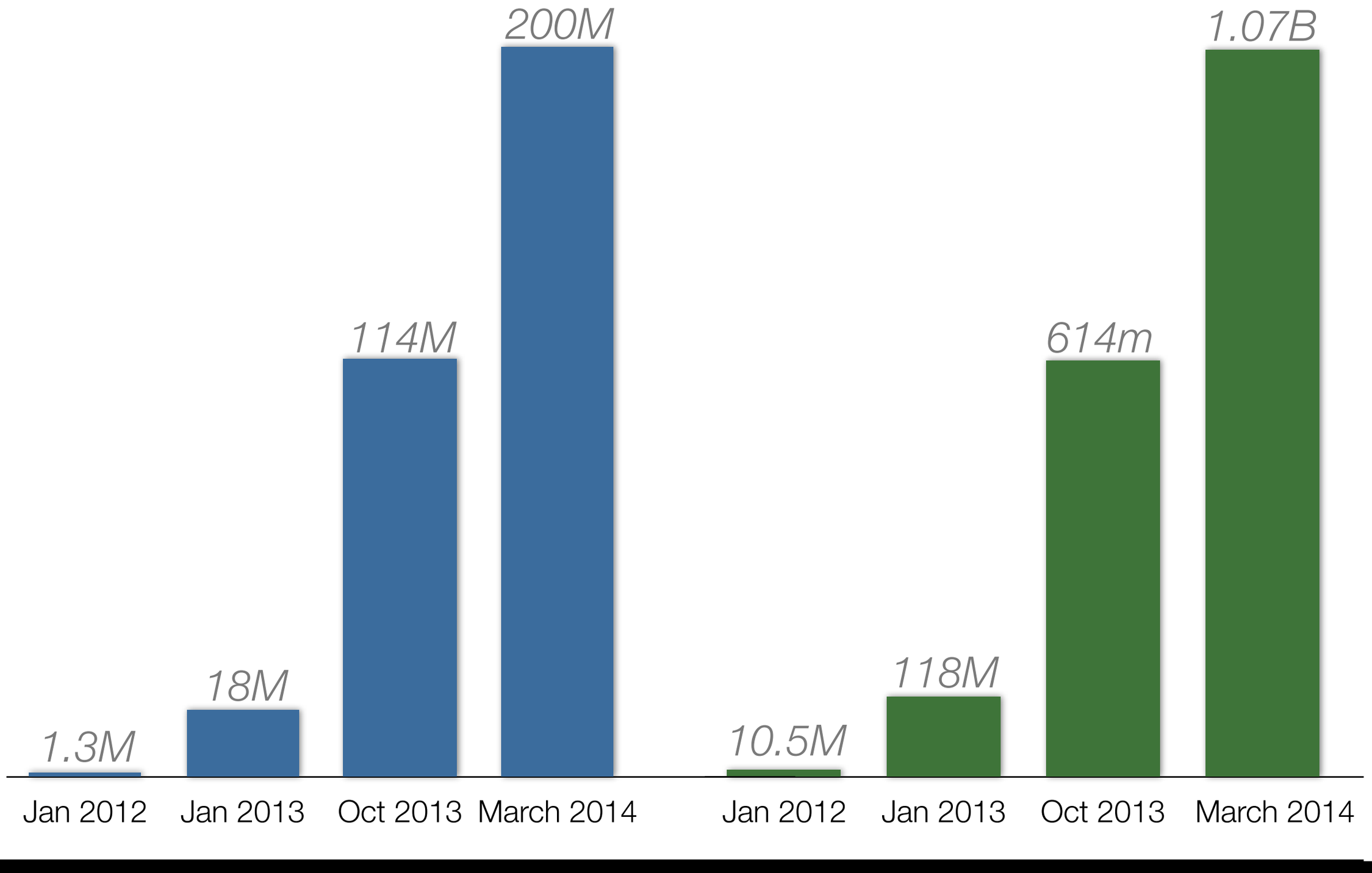
Who are you exactly?

Come to EmpireJS!

http://2014.empirejs.org

Why should I care about npm ?

Does two years of 1000% growth impress you?

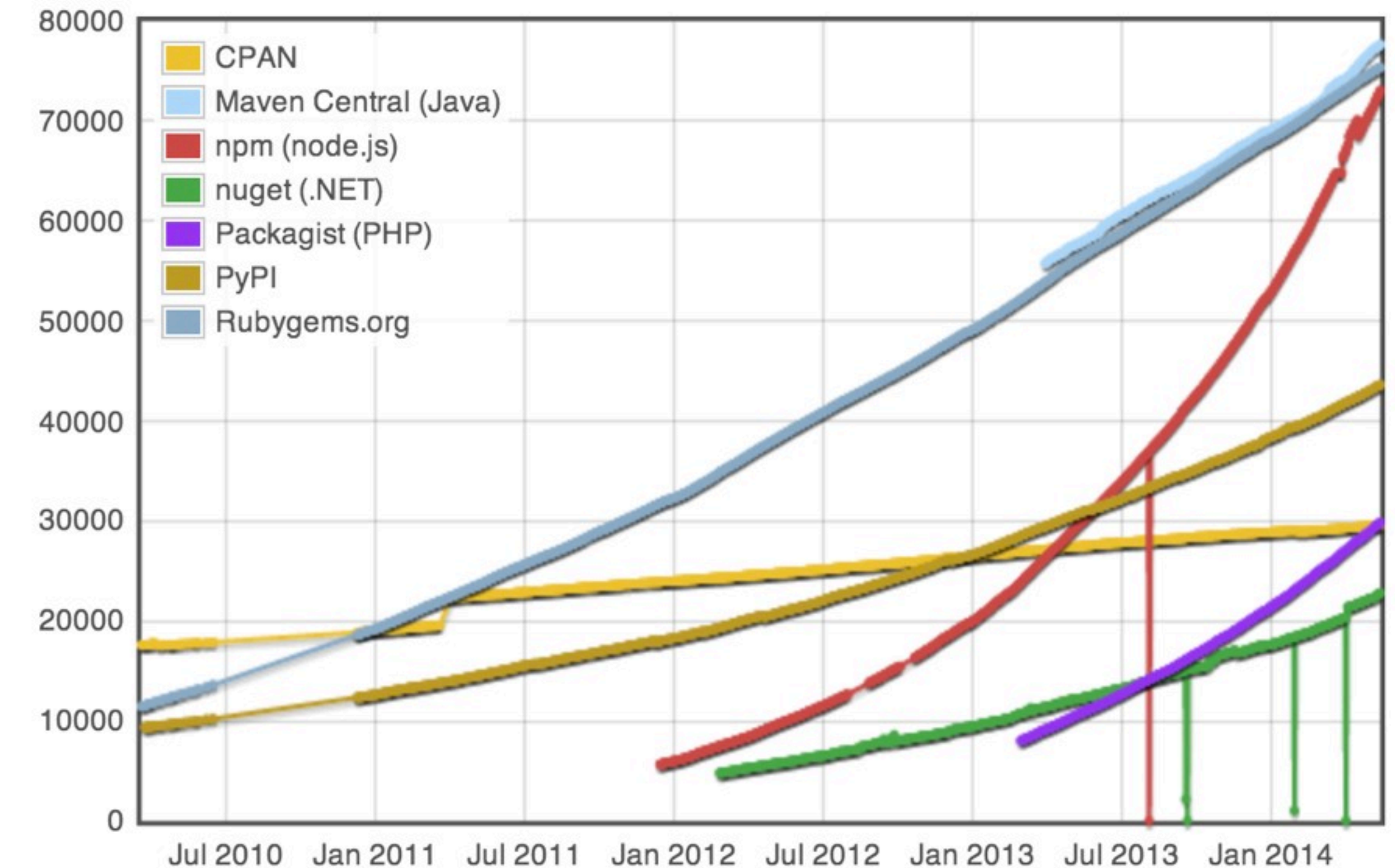


Module Downloads

Total Requests

Why should I care about npm ?

Because it will soon be the largest Open Source platform worldwide



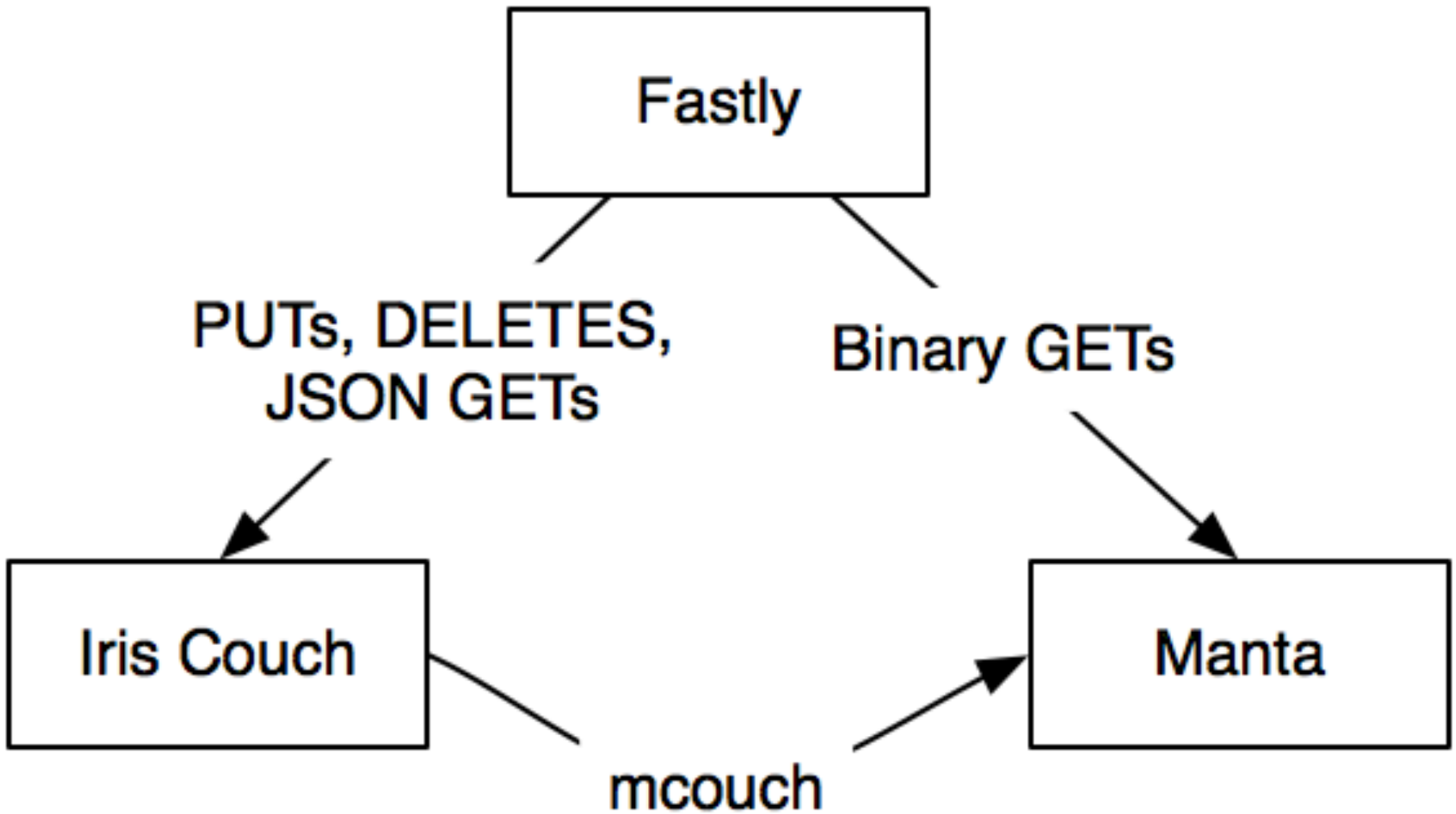
Source: modulecounts.com

CHESTER A. ARTHUR


npm
FALL
DOWN

Make npm not fall down.

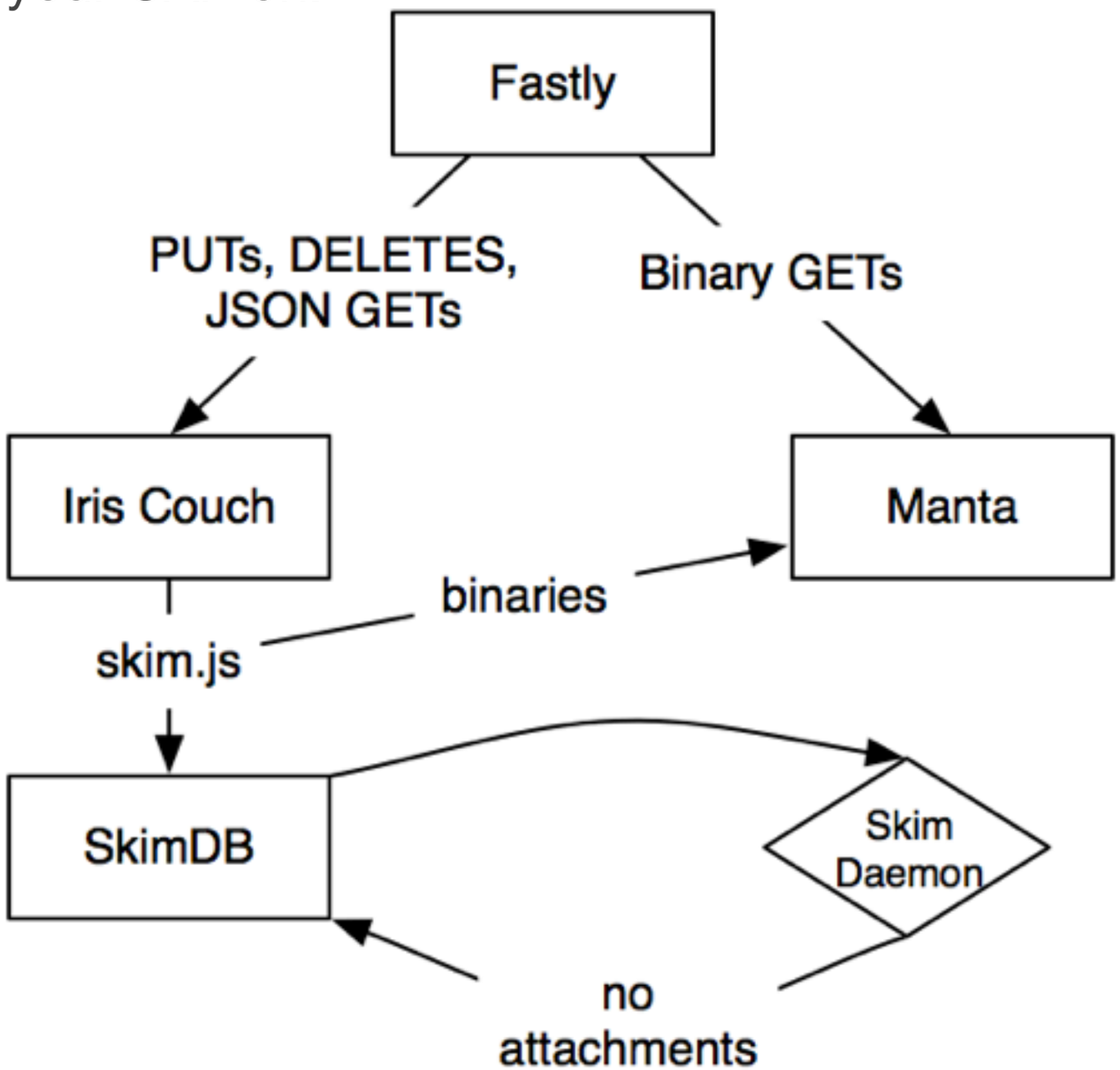
Make sure you've got your CAP on.



January 2014

Make npm not fall down.

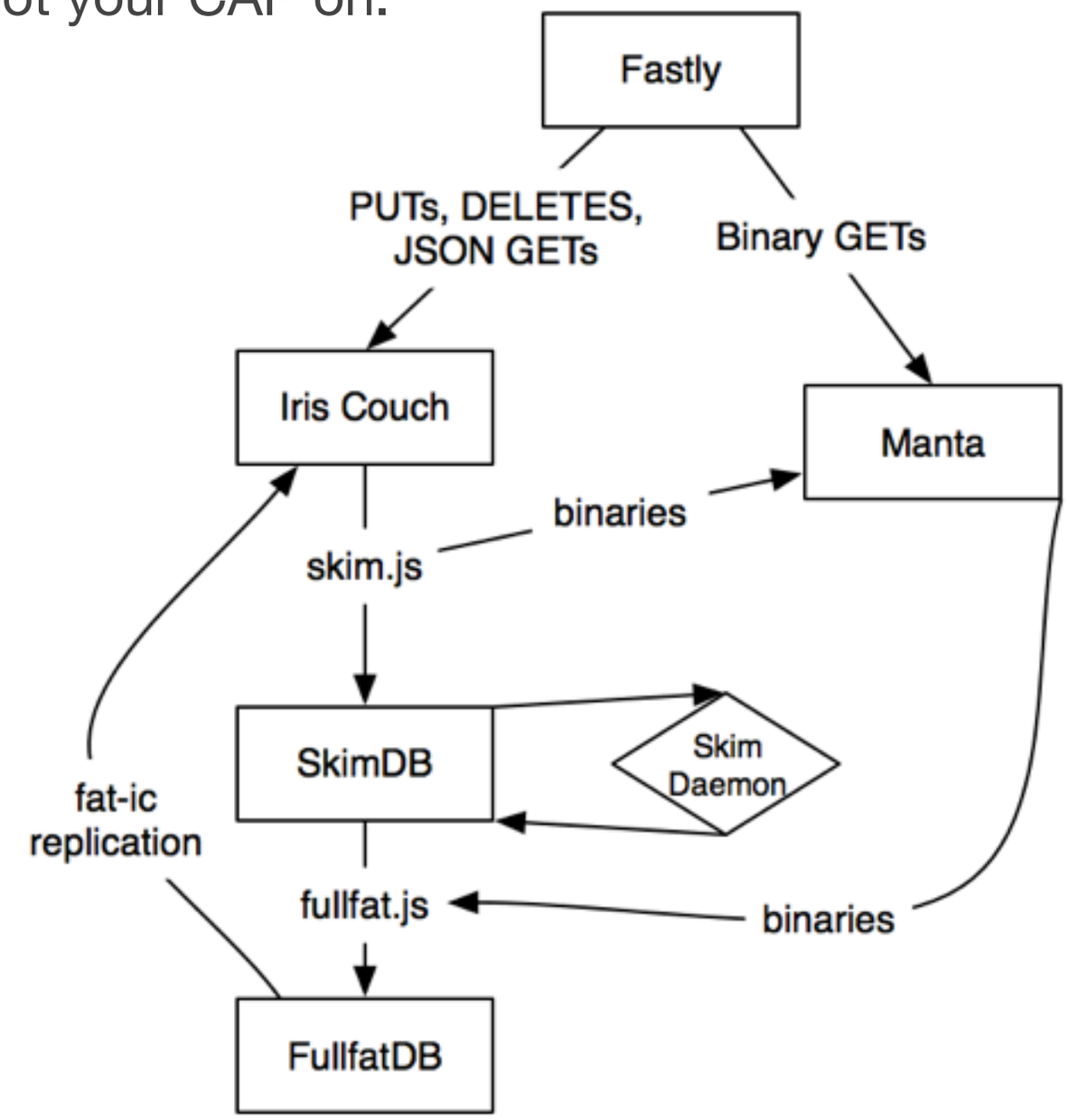
Make sure you've got your CAP on.



Transition One

Make npm not fall down.

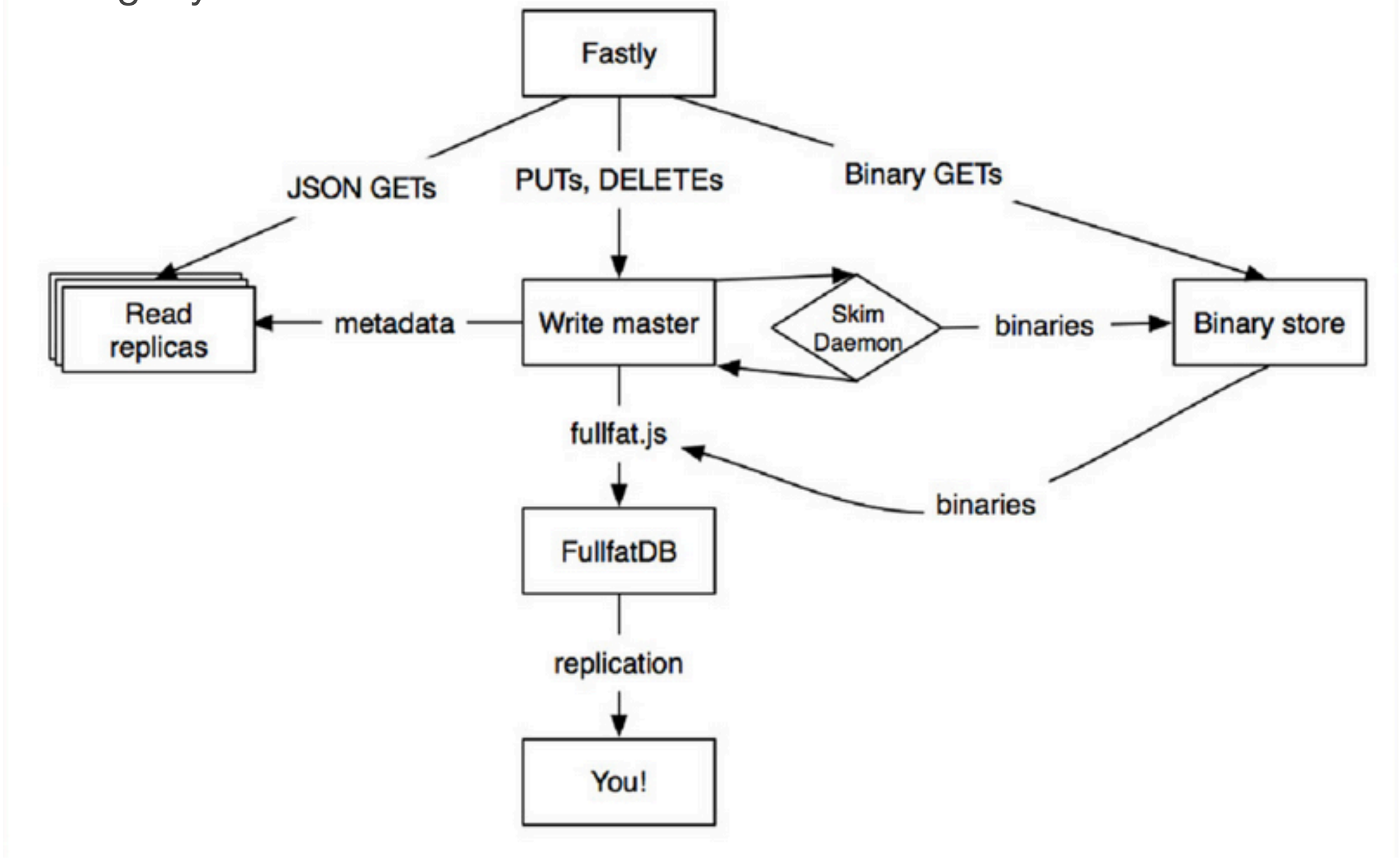
Make sure you've got your CAP on.



Transition Two

Make npm not fall down.

Make sure you've got your CAP on.



March 2014

This was good and bad for npm

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.**

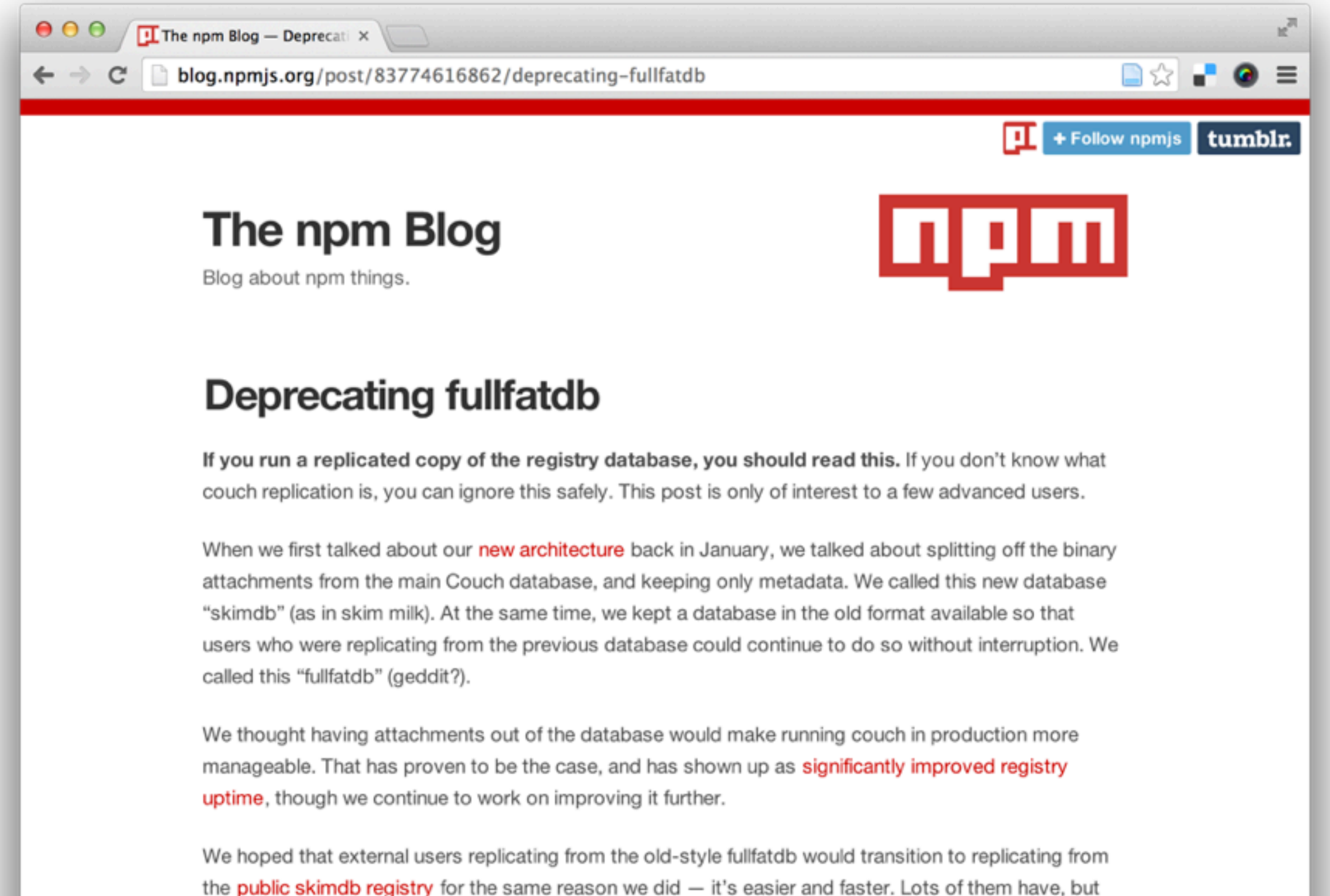
Good news!

- 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

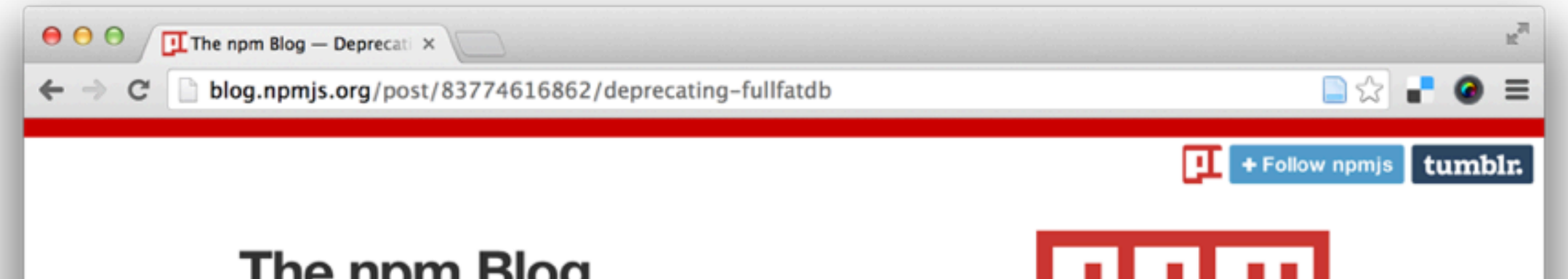
Probably not the best idea ... *seriously*



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.

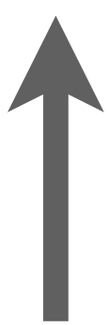

```
{
    _id: "winston",
    _rev: "182-f385b244a0f207487d56cc04ff836bec",
    name: "winston",
    description: "A multi-transport async logging library for Node.js",
  - dist-tags: {
      latest: "0.7.3"
    },
  - versions: {
    + 0.2.11: {…},
    + 0.3.3: {…},
    + 0.3.4: {…},
    + 0.3.5: {…},
    + 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",
```

npm with immutable documents

One attachment per document. **One version per document.**

express@4.0.0

express@4.0.0



$ npm publish

- 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

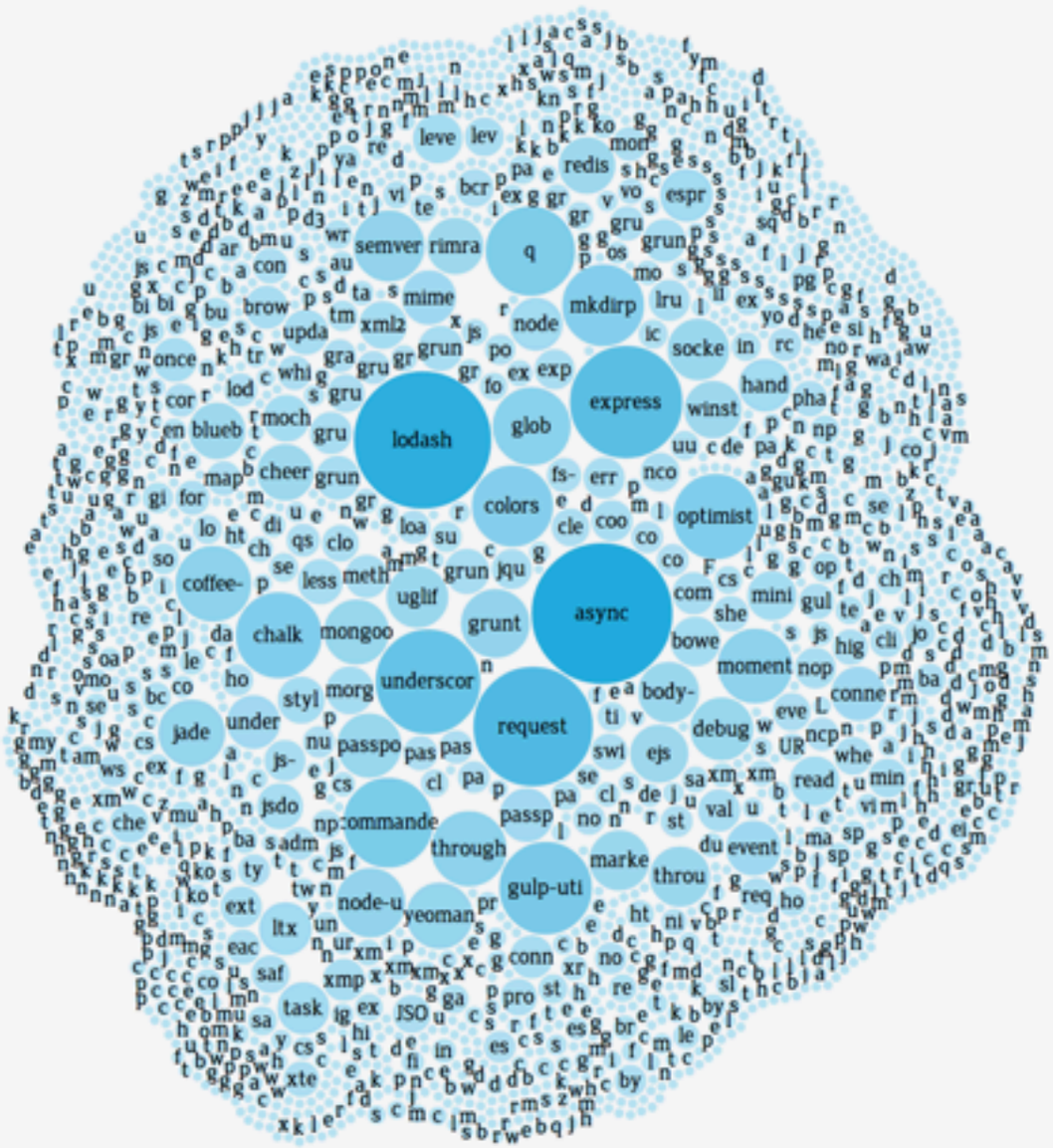
@jcrugzz

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.



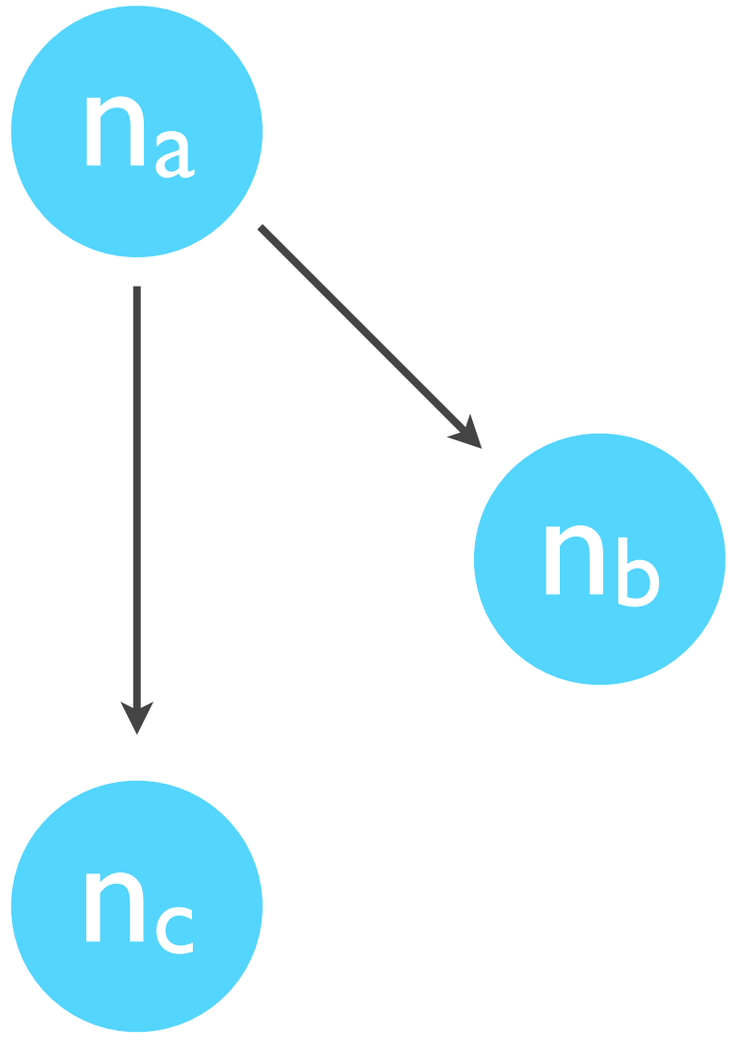
Dependency graphs

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 $\mathbf{n_A}$ to $\mathbf{n_B}$ if package A depends on package B.

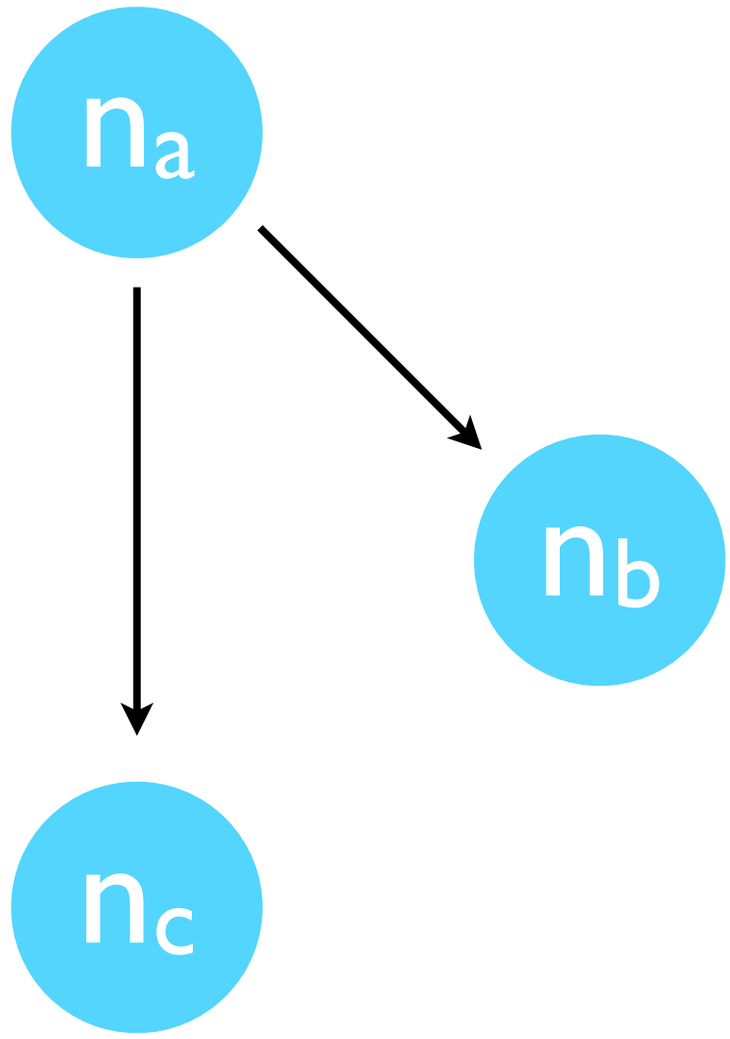
- Edges are colored by dependency type: `dependencies` or `devDependencies`



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.**

MIND. EXPLOSION.

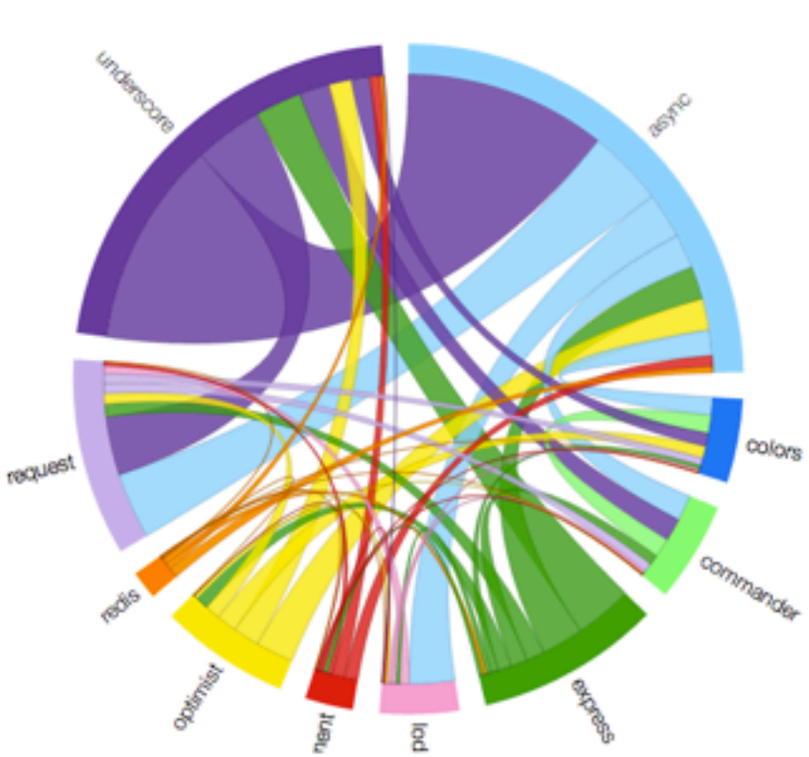
Actually.
It’s not so bad.

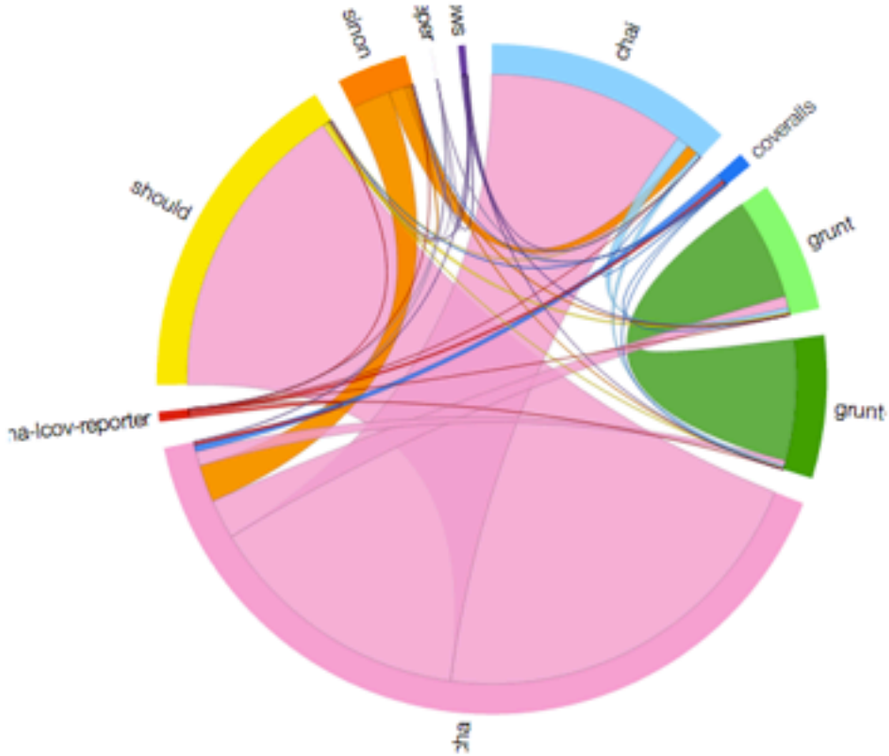
Dependency graphs

Ok, so what is this useful for?

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

First: Codependencies





Codepdencies?

Well, technically co(*)dependencies.

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

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

<3 CouchDB!

Codepdencies?

Thanks CouchDB!

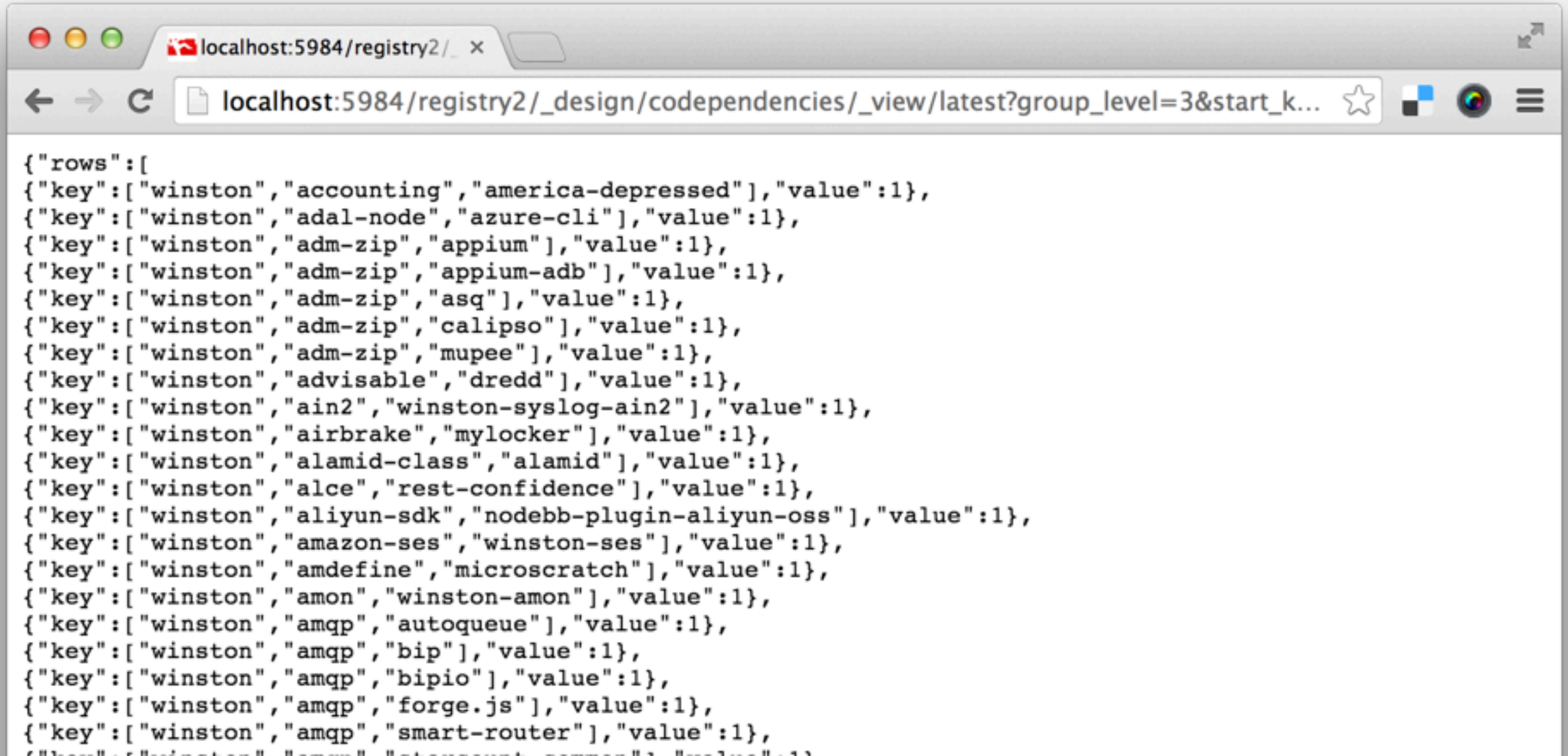
- We can get all of this from a **simple CouchDB view.**

```
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)
    }
  }
}
```

Codepdencies?

Thanks CouchDB!

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



Codepdencies

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]
```

Codepdencies

Understanding codependencies through winston

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

	asyn…	expr…	opti…	requ…	unde…
asyn…	0.0000	553.0000	534.0000	837.0000	1359.0000
expr…	553.0000	0.0000	314.0000	365.0000	648.0000
opti…	534.0000	314.0000	0.0000	335.0000	448.0000
requ…	837.0000	365.0000	335.0000	0.0000	786.0000
unde…	1359.0000	648.0000	448.0000	786.0000	0.0000

Codepdencies

Understanding codependencies through winston

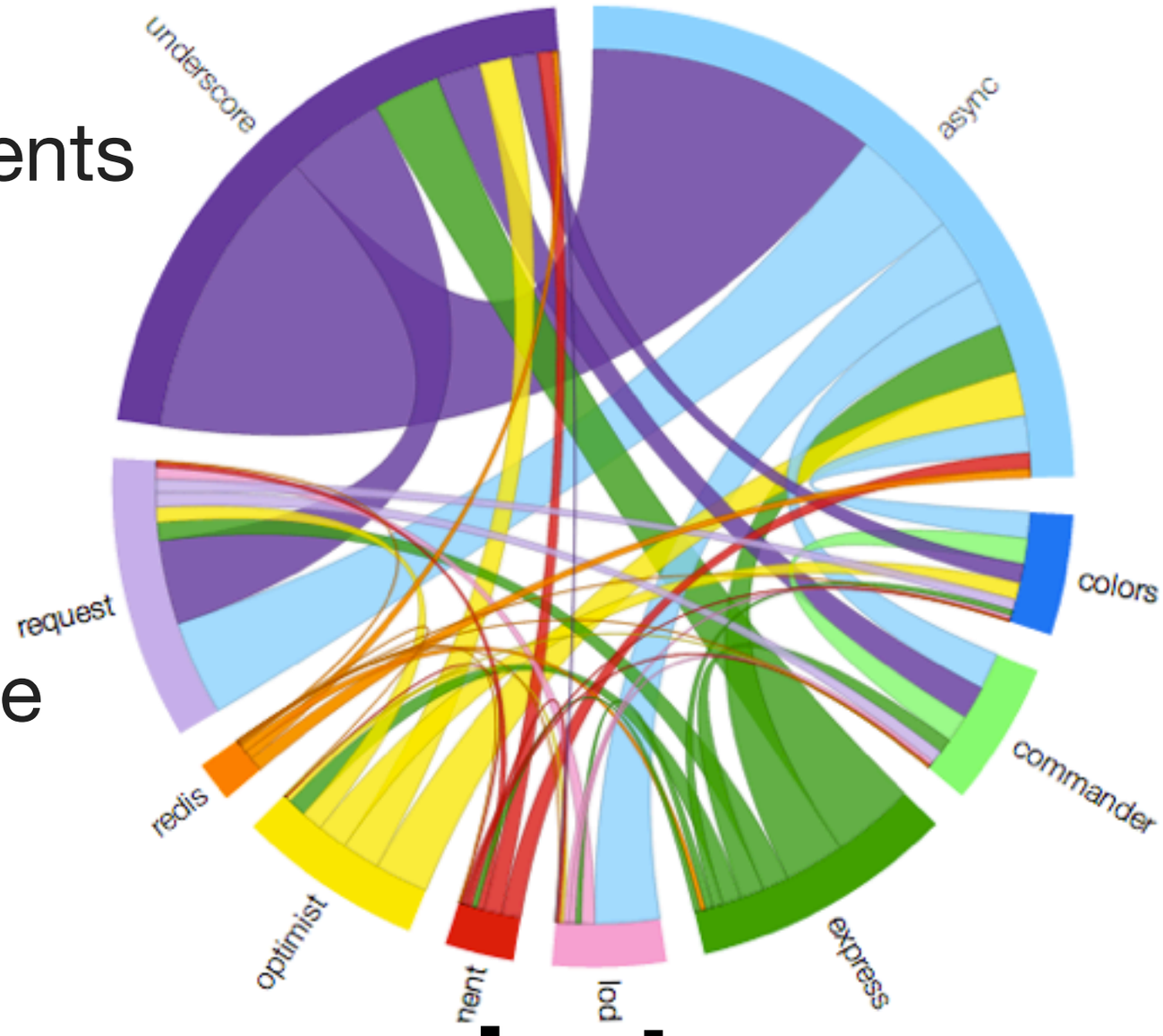
- Now we need to weight the matrix based on the overall appearance of these codependencies

```
240 | async      | 0.2761
207 | underscore | 0.2382
163 | express    | 0.1875
133 | request    | 0.1530
126 | optimist   | 0.1449
869 total
```

Codepdencies

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.



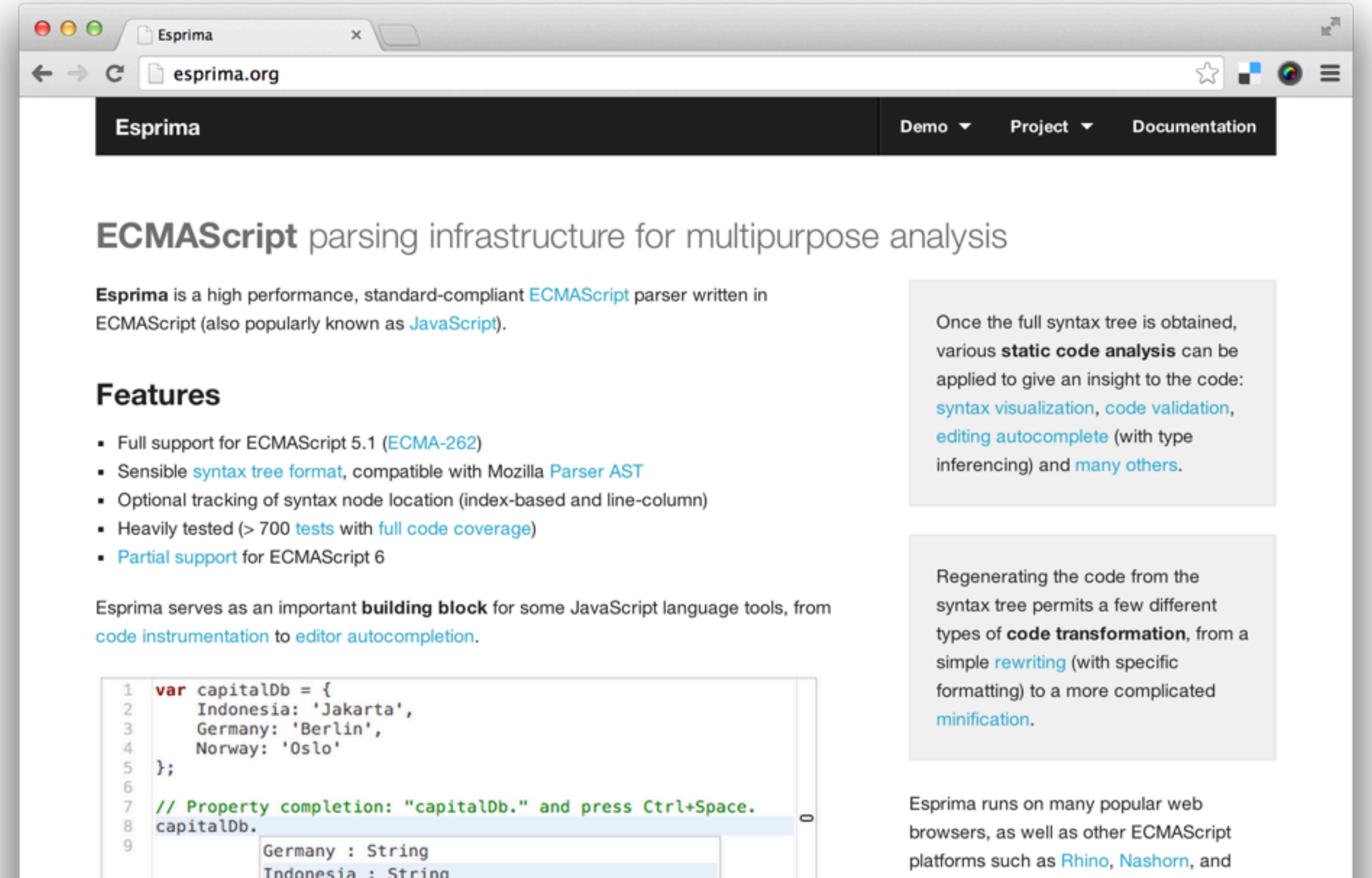
PAUSE FOR DEMO

I should use static analysis with npm

ANALYZE ALL THE THINGS!

I should use static analysis with npm

How the what exactly?



I should use static analysis with npm

Why the what precisely?

compstat.pdf

file:///Users/charlie/Downloads/compstat.pdf

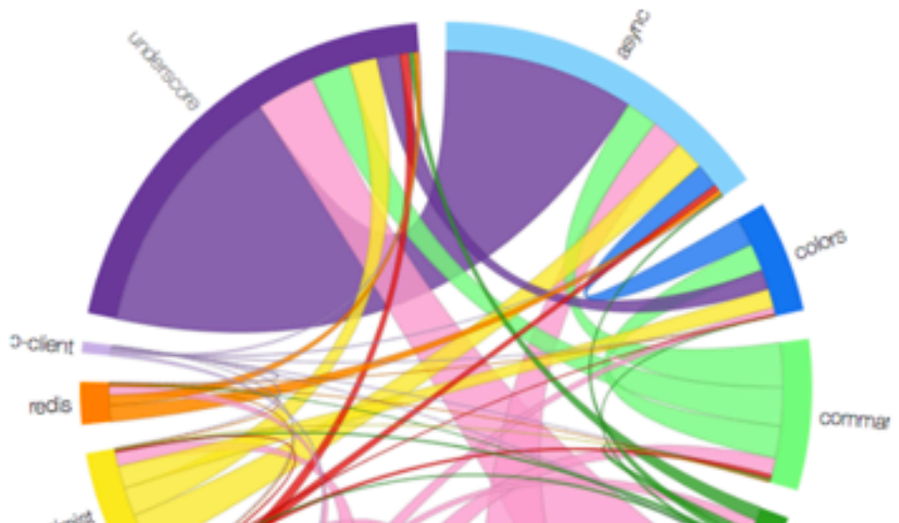
Toward decreasing mean decision-time in Open Source with com(STAT)^2

Charlie Robbins, Jane Kim, William McAuliff
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 modules(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



I should use static analysis with npm

Why the what precisely?

SMALL GROUP

OF 8 EXPERTS

I should use static analysis with npm

Why the what precisely?

MOST EFFECTIVE METHODS USED MOST INFREQUENTLY

Frequency

Never | Somewhat Infrequently | Somewhat Frequently | Always

Open A Github Issue	30% Frequently
Send a Tweet	30% Somewhat Frequently
Mention users on Github	40% Somewhat Frequently
Mention users on Twitter	40% Somewhat Infrequently
Write a blog post	40% Never
Email a mailing list	50% Never
Contact individual users	30% Somewhat Frequently

Open A Github Issue	Effective 40%
Send a Tweet	Somewhat Effective 40%
Mention users on Github	Effective 30%
Mention users on Twitter	Somewhat Ineffective 20%
Write a blog post	Ineffective 20%
Email a mailing list	Somewhat Ineffective 30%
Contact individual users	Very Effective 40%

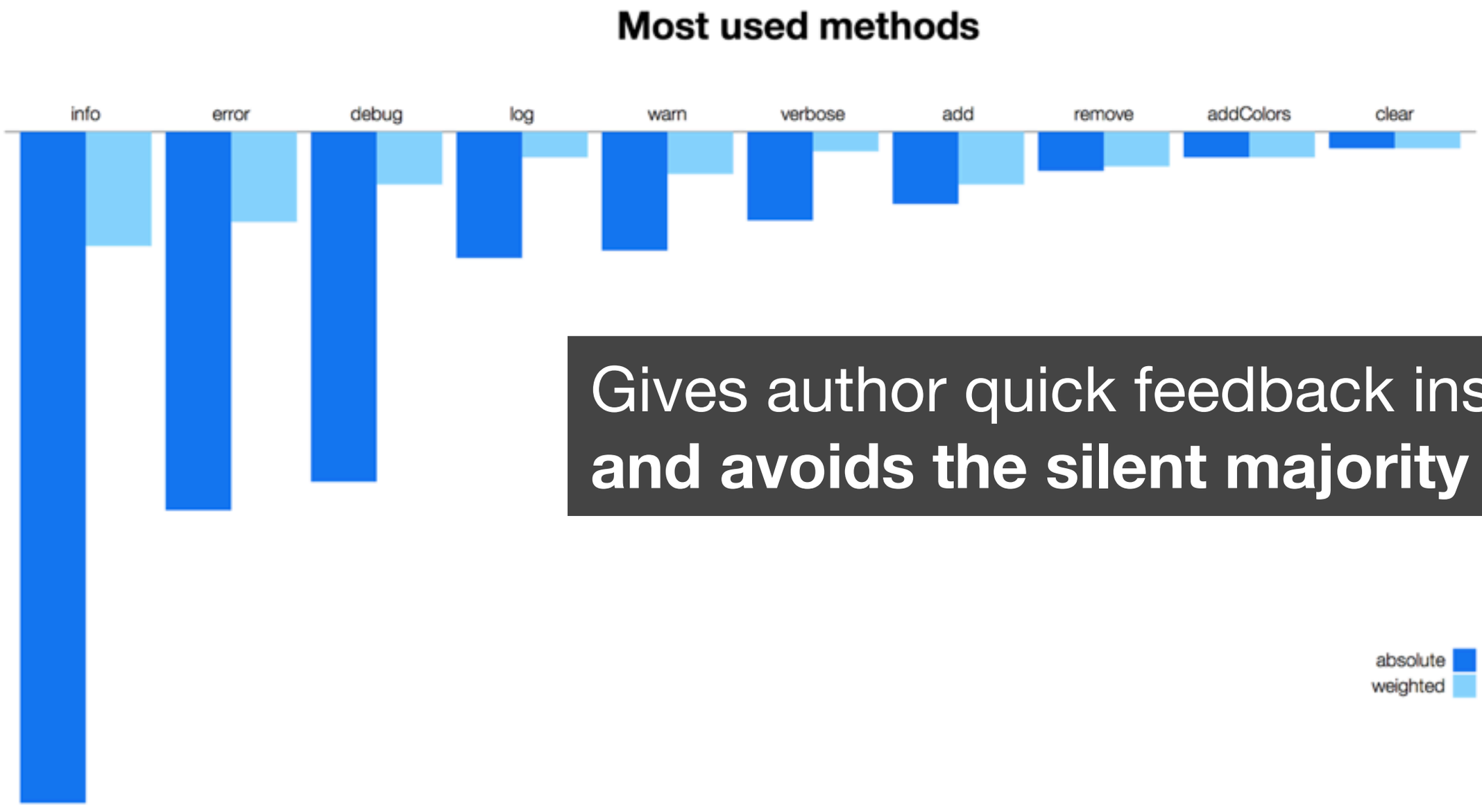
Very Ineffective | Somewhat Ineffective | Somewhat Effective | Very Effective

Effectiveness

I should use static analysis with npm

GET TO THE POINT ALREADY CHARLIE!

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



Gives author quick feedback instantly
and avoids the silent majority

PAUSE FOR OPEN SOURCE

Using private modules with npm

Problem. Solved. Headshot.

Using private modules with npm

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"
}
```

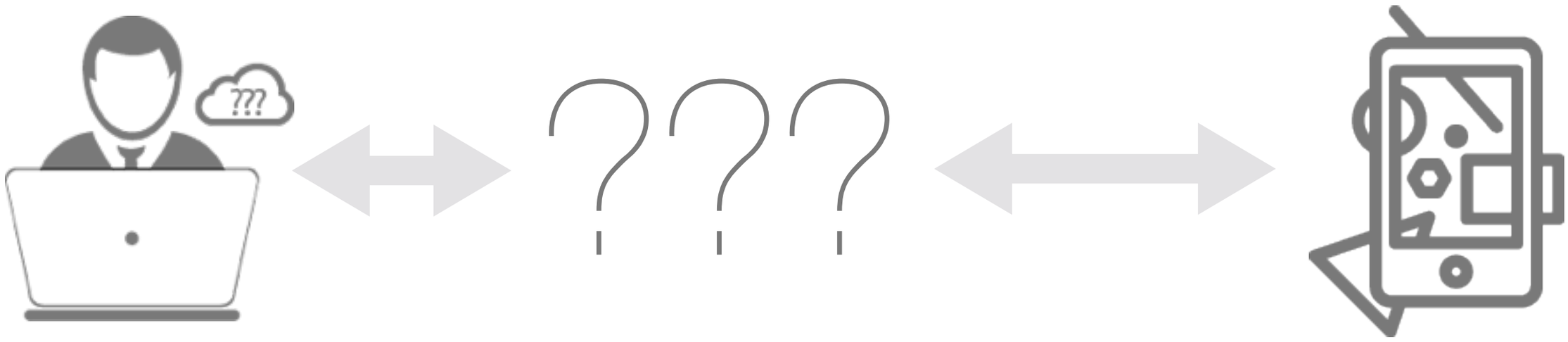
GLOBAL MODULES CANNOT DEPEND ON SCOPED MODULES!

WARNING: shameless plug

for  nodejitsu

I will be brief. I promise.

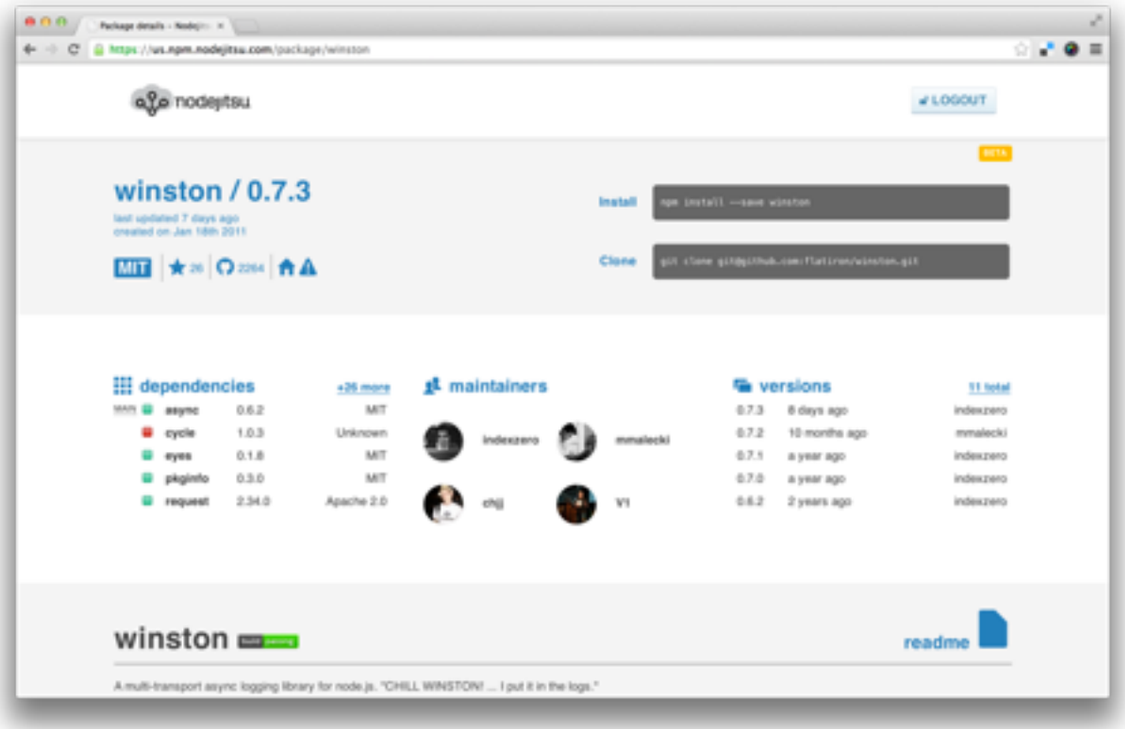
What **problem do we solve** for npm?



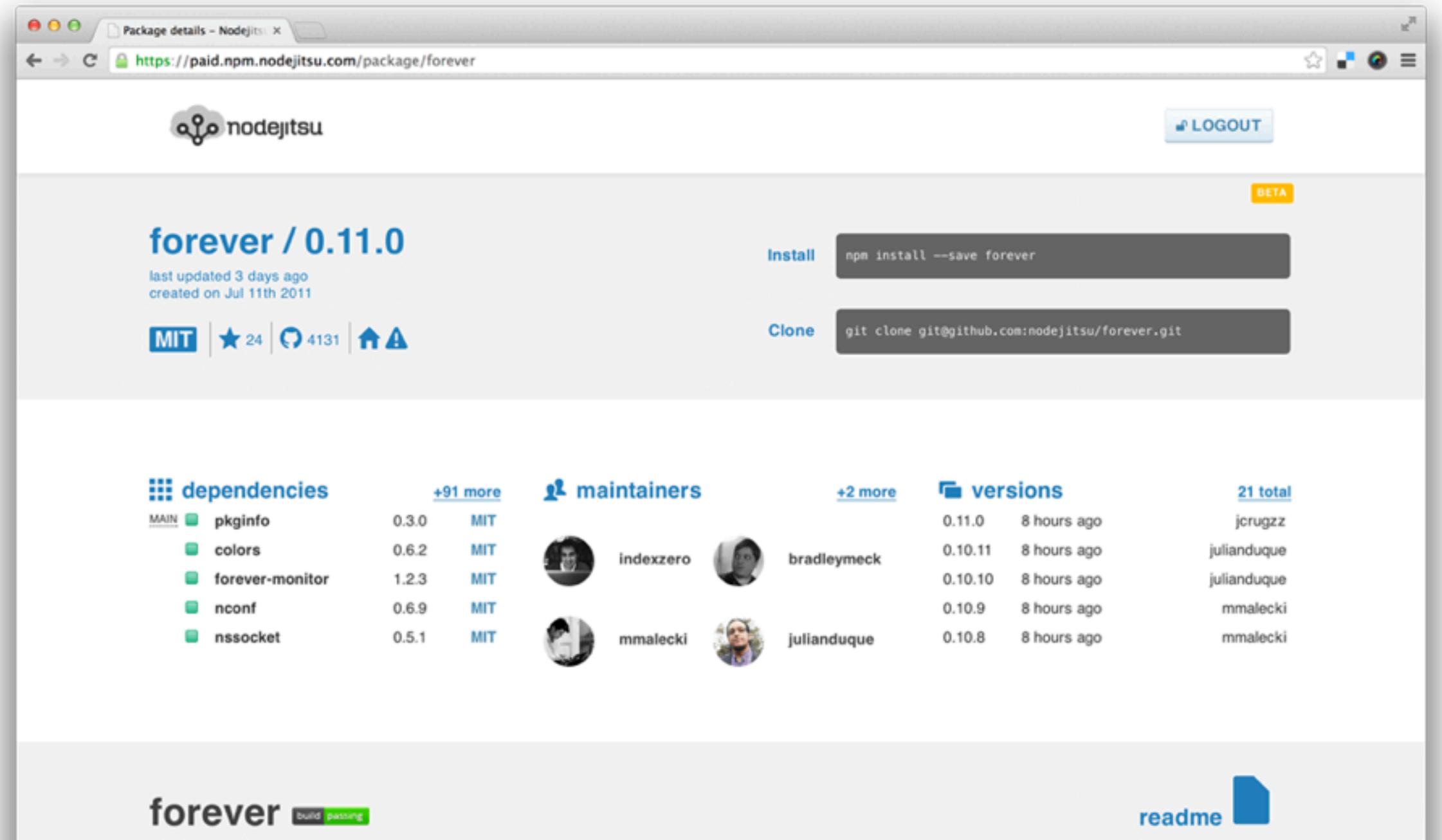
Developers have **questions** about their **applications**

We increase value & avoid wasted time by answering these questions

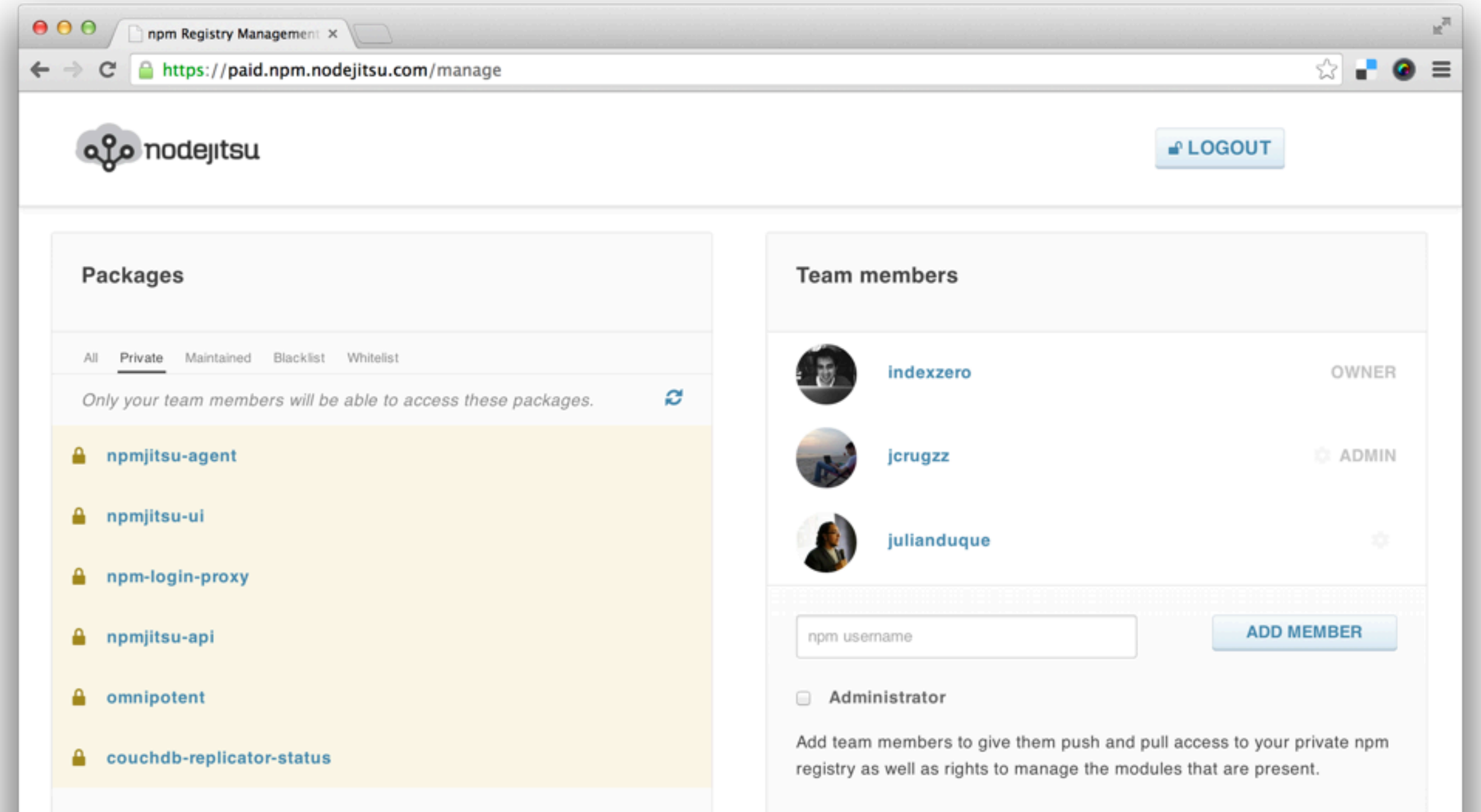
- *Where do I **store my private code**?*
- *Is my Node.js application **secure**? Against what vulnerabilities?*
- *How do I **deploy** my application?*
- *Am I at **legal risk** because of Open Source licenses?*
- ***What module should I use** to do <feature> in my app?*
- *Is this Open Source module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



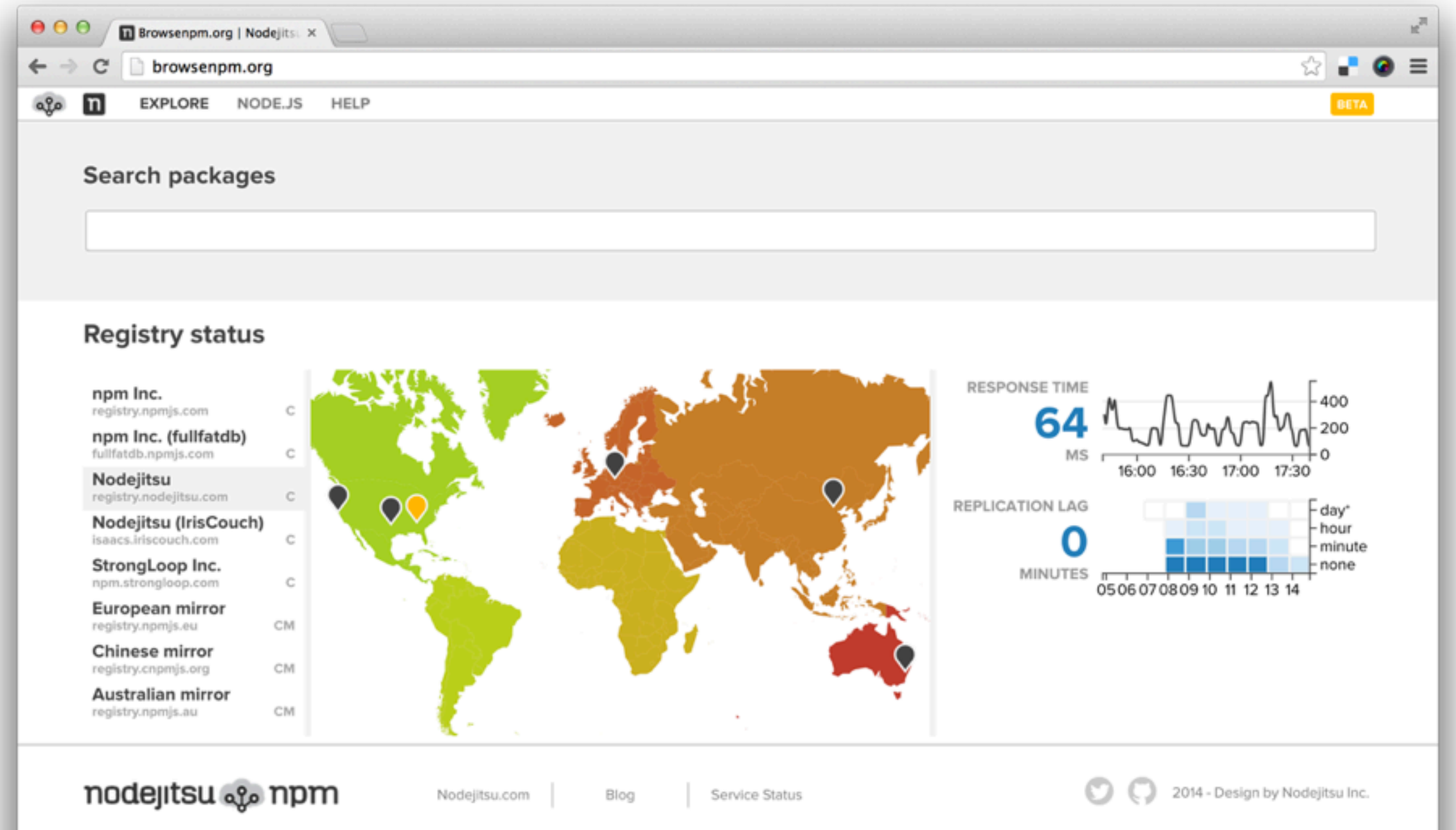
Github for production Javascript



Manage your team.
Manage your code.



Try it today
nodejitsu.com/signup



FIN