GRAPH THEORY

MIND. EXPLOSION.

Who are you exactly?

And why should I pay attention?

Founder & CEO at

I'm on the Internet!

@indexzero

AND RECENTLY...

LITERALLY A MASTER

OF HCI

HUMAN-COMPUTER INTERACTION

TERM POPULARIZED IN THE 1980s

Human–computer interaction (HCI) involves the study, planning, design and uses of the interaction between people (users) and computers. It is often regarded as the intersection of computer science, behavioral sciences, design, media studies, and several other fields of study.

HCI FIGHTS FOR THE USER

WELL ... SORT OF ...

HCI & Static Analysis?

Why the what precisely?

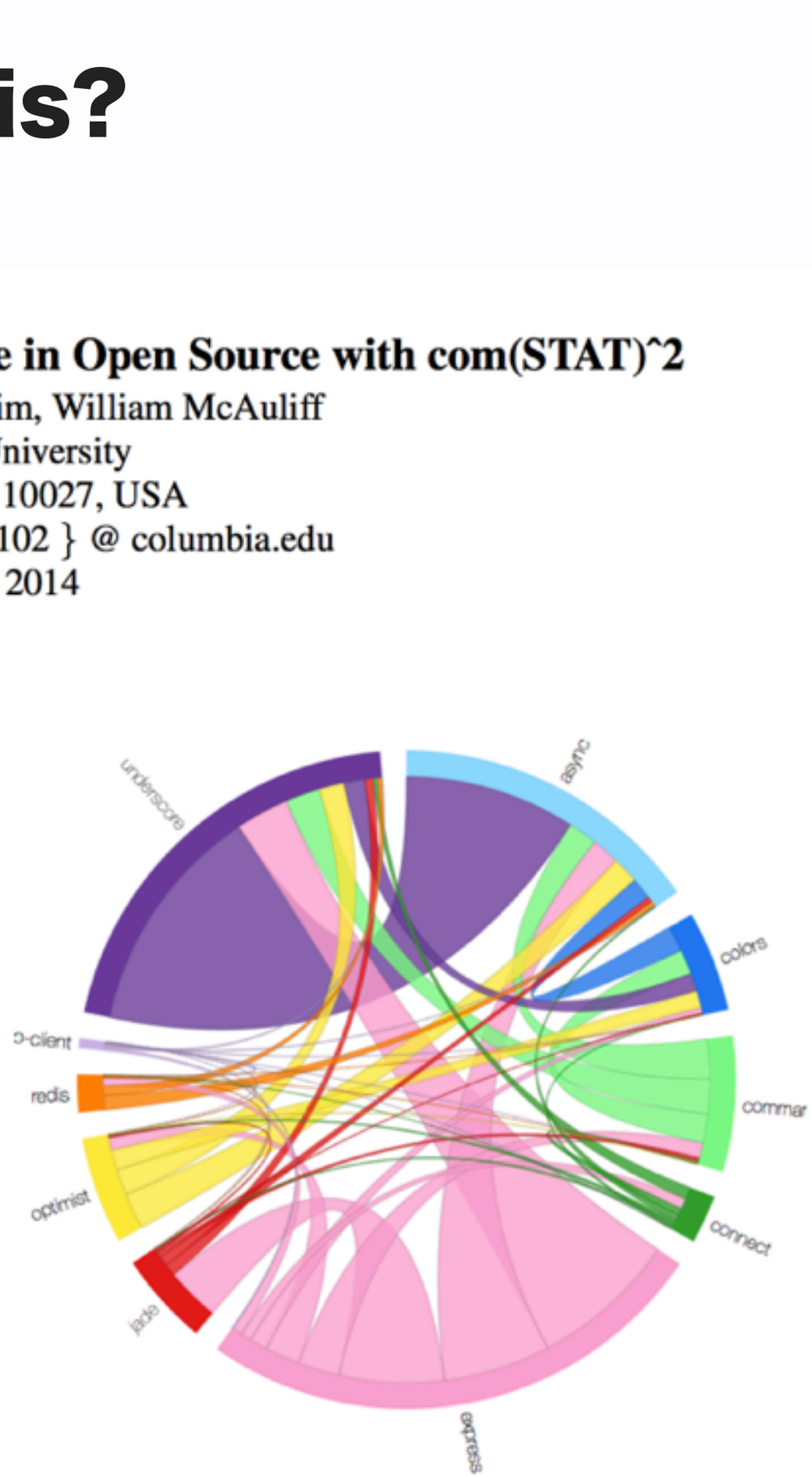
Toward decreasing mean decision-time in Open Source with com(STAT)^2

Charlie Robbins, Jane Kim, William McAuliff
Columbia University
New York, NY, 10027, USA
{ cjr2139, jk3316, wom2102 } @ columbia.edu
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 presents a COMprehensive STATistical STATic analysis interface, com(STAT)^2, that aims to decrease the mean decision time for module authors and evaluates it on well-known authors in the popular Javascript environment of Node.js and npm (Node Packaged Modules).



1 Establishing the Expert Problem Domain

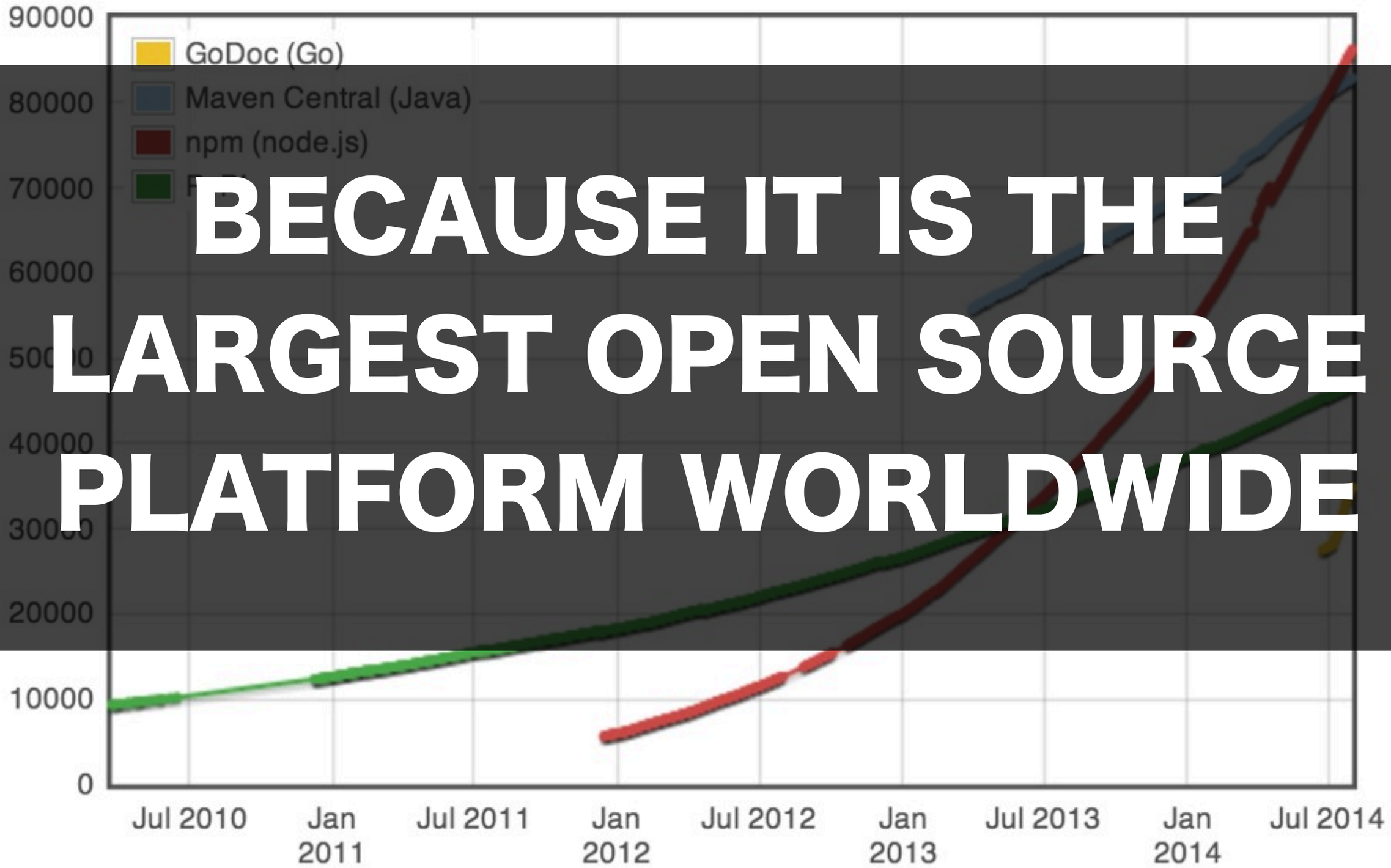
HCI

WITH 

I CAN HAZ USER?

Why is npm a worthy HCI topic?

Because reasons I guess maybe?



BECAUSE IT IS THE LARGEST OPEN SOURCE PLATFORM WORLDWIDE

Source: modulecounts.com

Why is npm a worthy HCI topic?

How soon exactly?

AS OF 06/26/14

npm	Maven
80,353	80,676
-323	+323
173 / day	75 / day

ROUGHLY BY TUESDAY.

YUP.

PLEASE. TELL ME ABOUT YOUR RESEARCH STUDY

Why is npm a worthy HCI topic?

Why the what precisely?

SMALL GROUP

OF 8 EXPERTS

Likert Scales.

Seriously, why does this need a name?

Likert

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

Likert

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

Why is npm a worthy HCI topic?

Why the what precisely?

MOST EFFECTIVE METHODS USED MOST INFREQUENTLY

MODULE AUTHORS AND DEVELOPERS HAVE QUESTIONS ABOUT MODULES BEING DEVELOPED

HCI TRON MUST KNOW

HOW DO WE DO BETTER?

GRAPHS

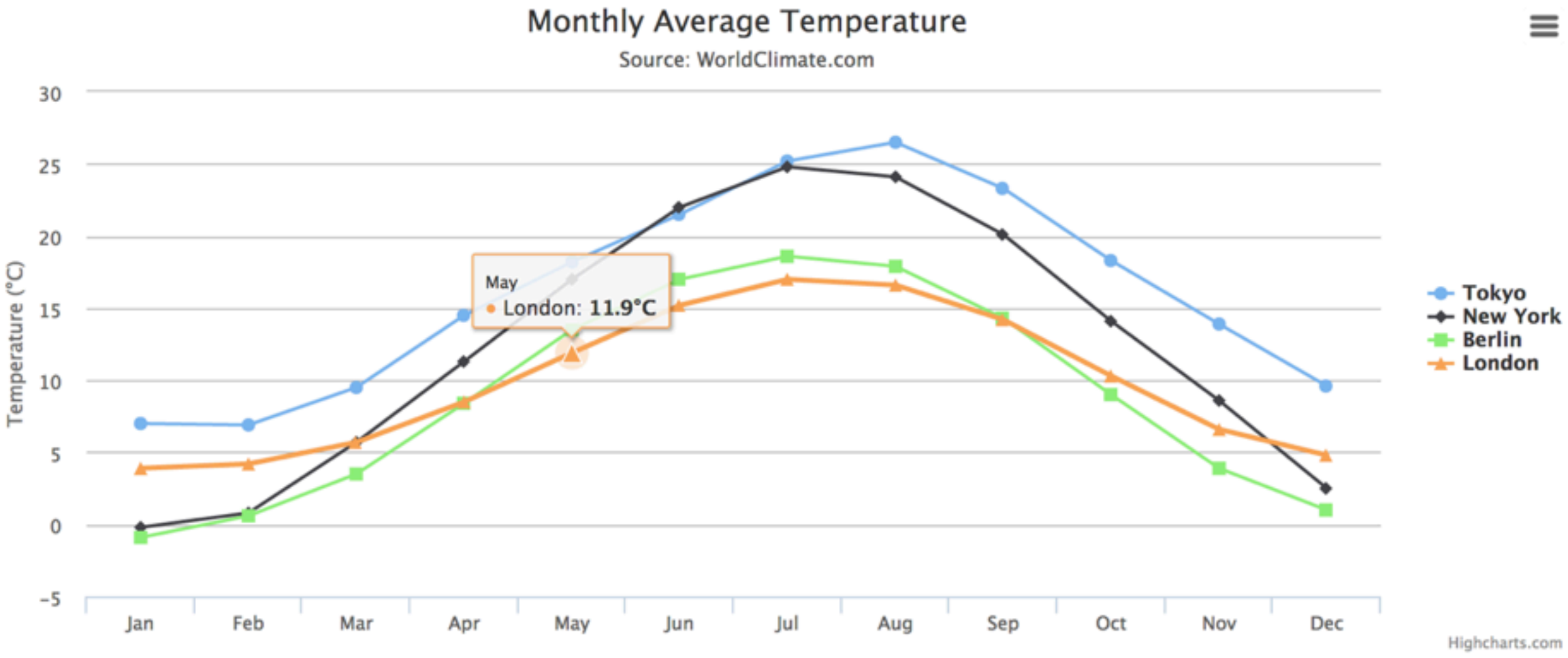
RULE

EVERYTHING

AROUND

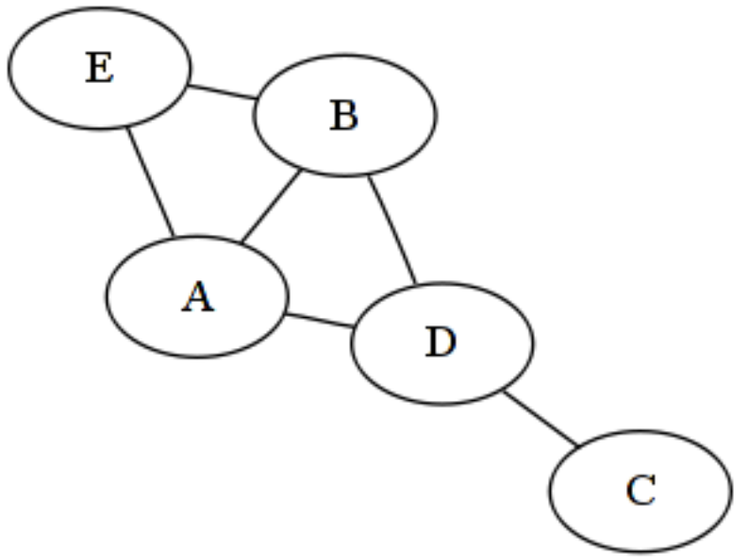
ME

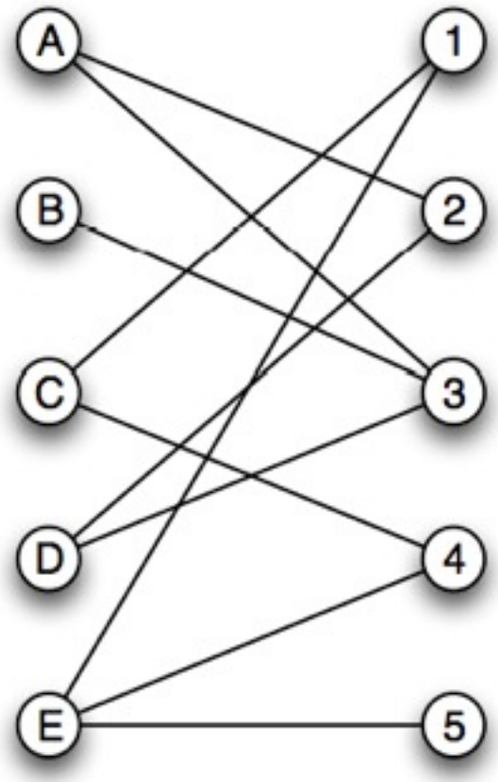
NOT THIS KIND

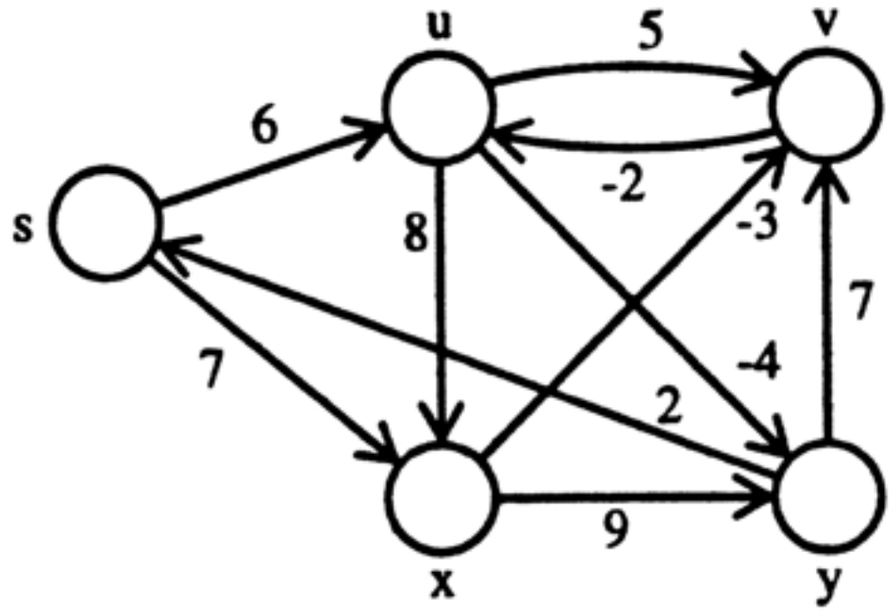


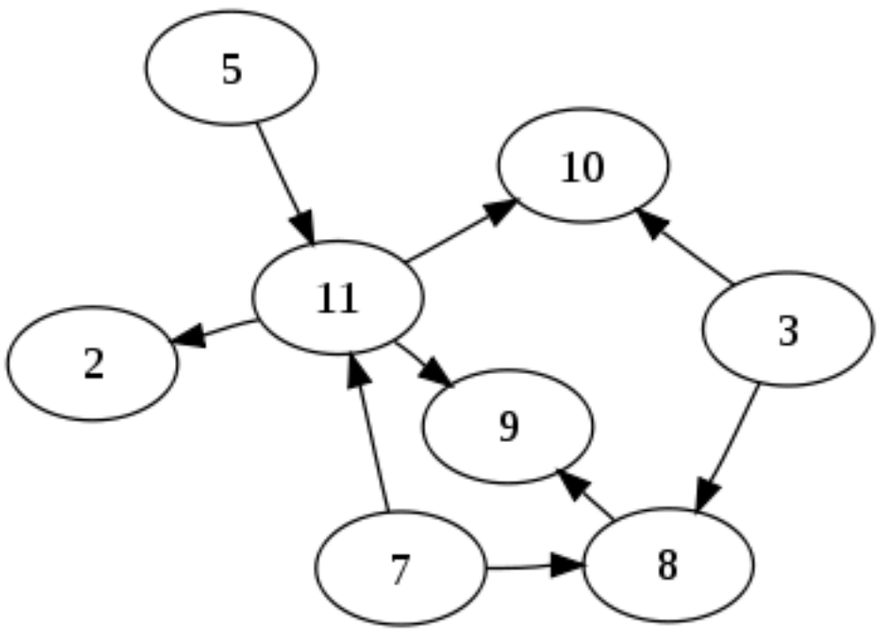
OF GRAPH

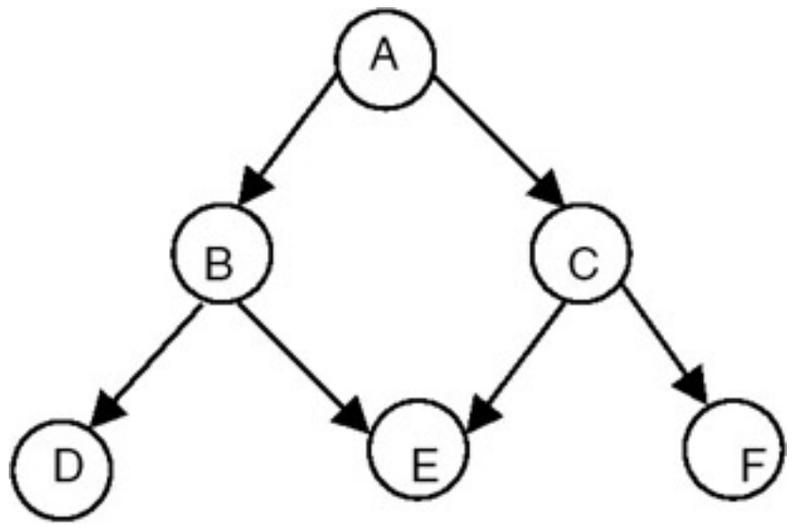
THESE KINDS





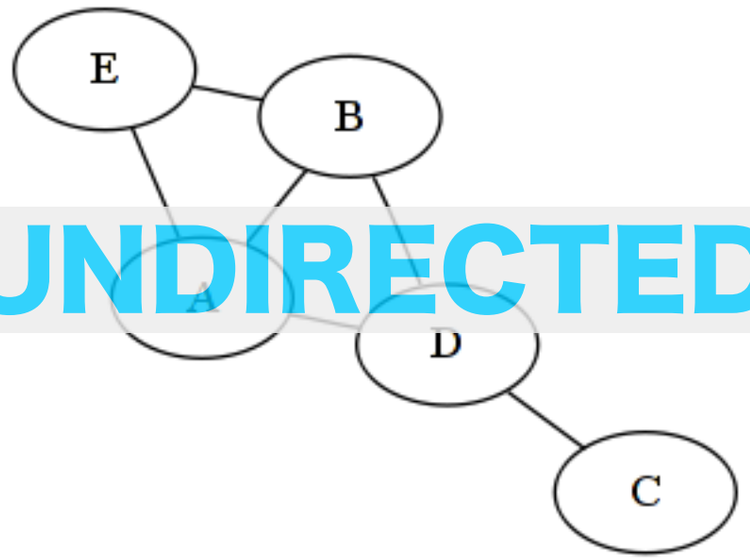




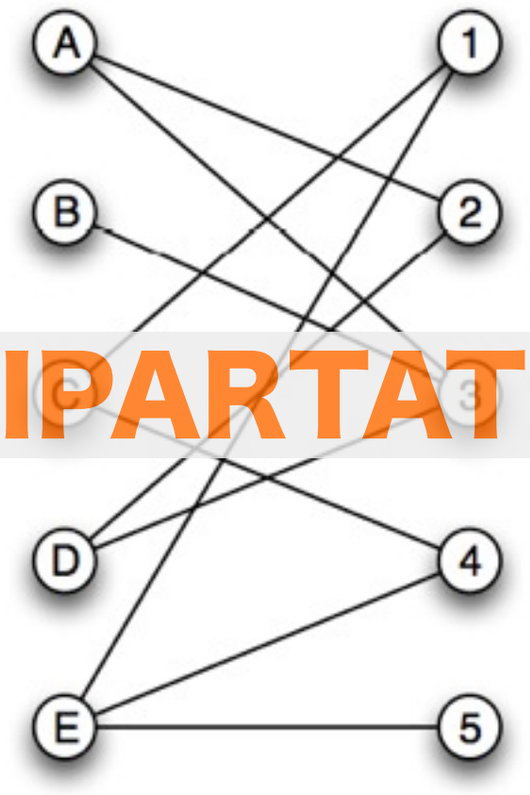


OF GRAPHS

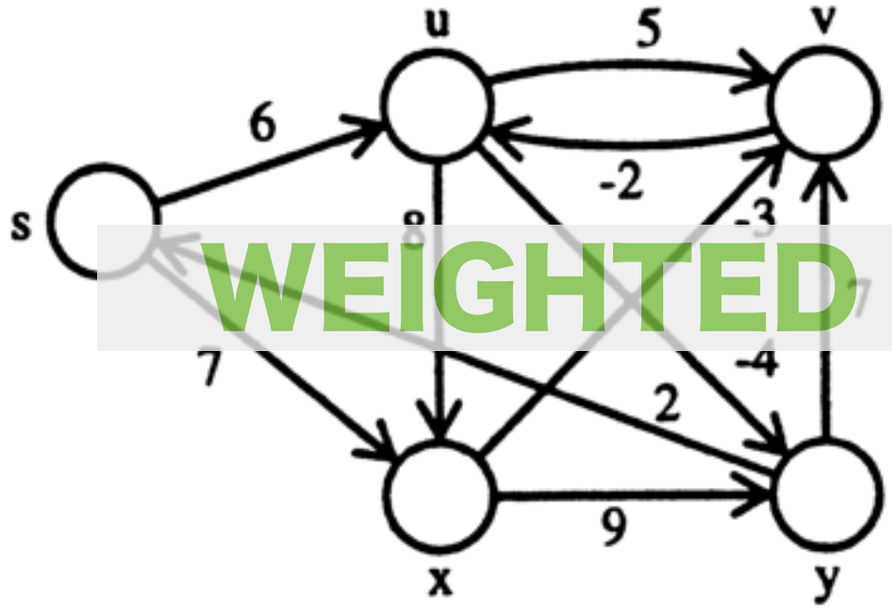
ALL OF THESE GRAPHS



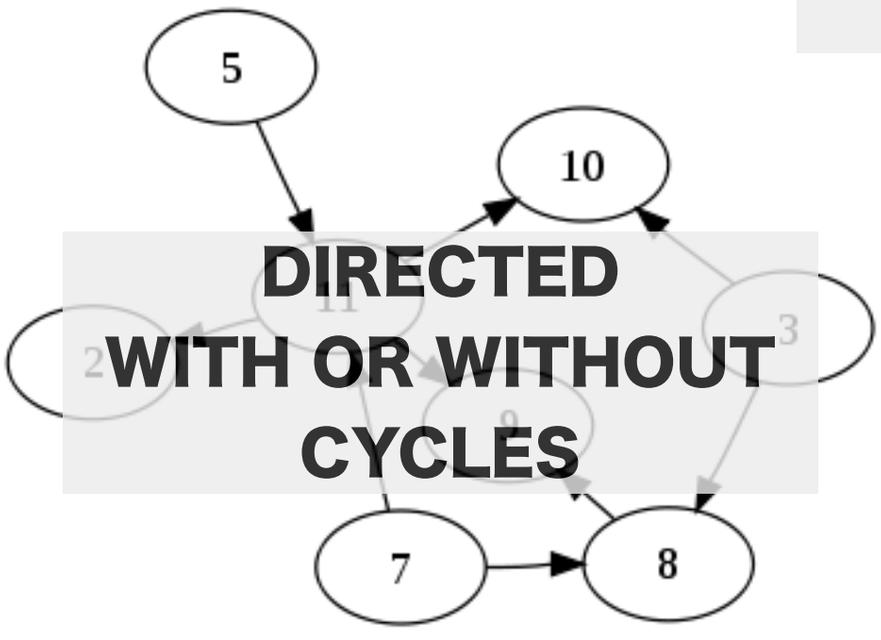
UNDIRECTED



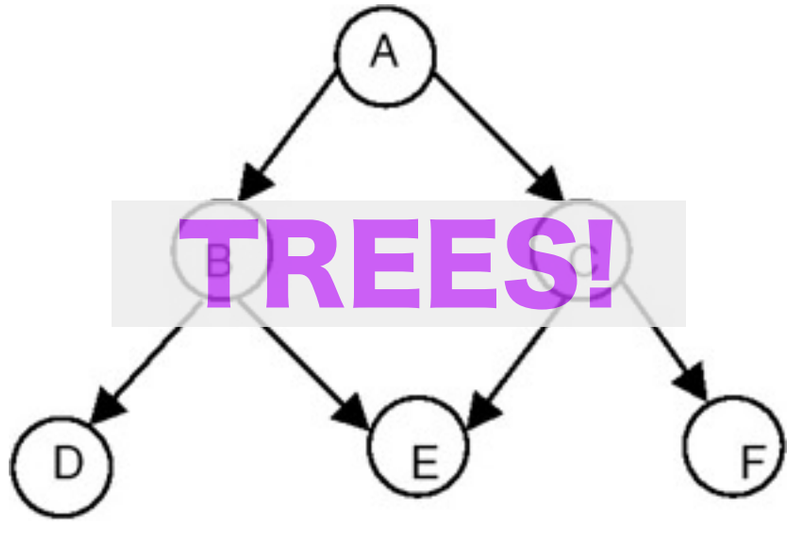
BIPARTATE



WEIGHTED



DIRECTED
WITH OR WITHOUT
CYCLES



TREES!

ARE DIFFERENT

TRANSPORATION NETWORKS

INFORMATION NETWORKS

MOLECULAR CHEMISTRY

WIRELESS NETWORKS

MAJOR LEAGUE BASEBALL

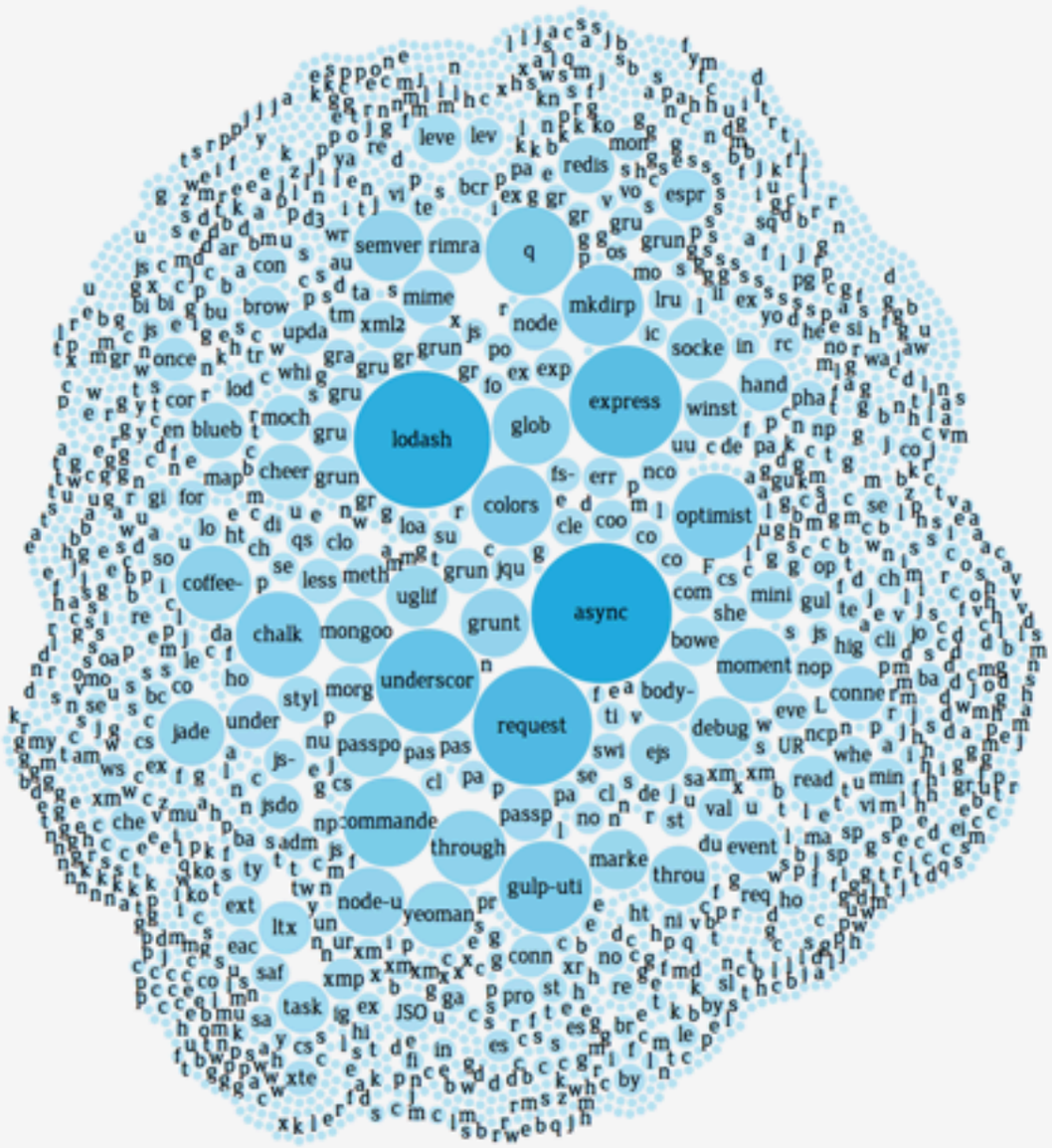
DEPENDENCY MANAGEMENT

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.



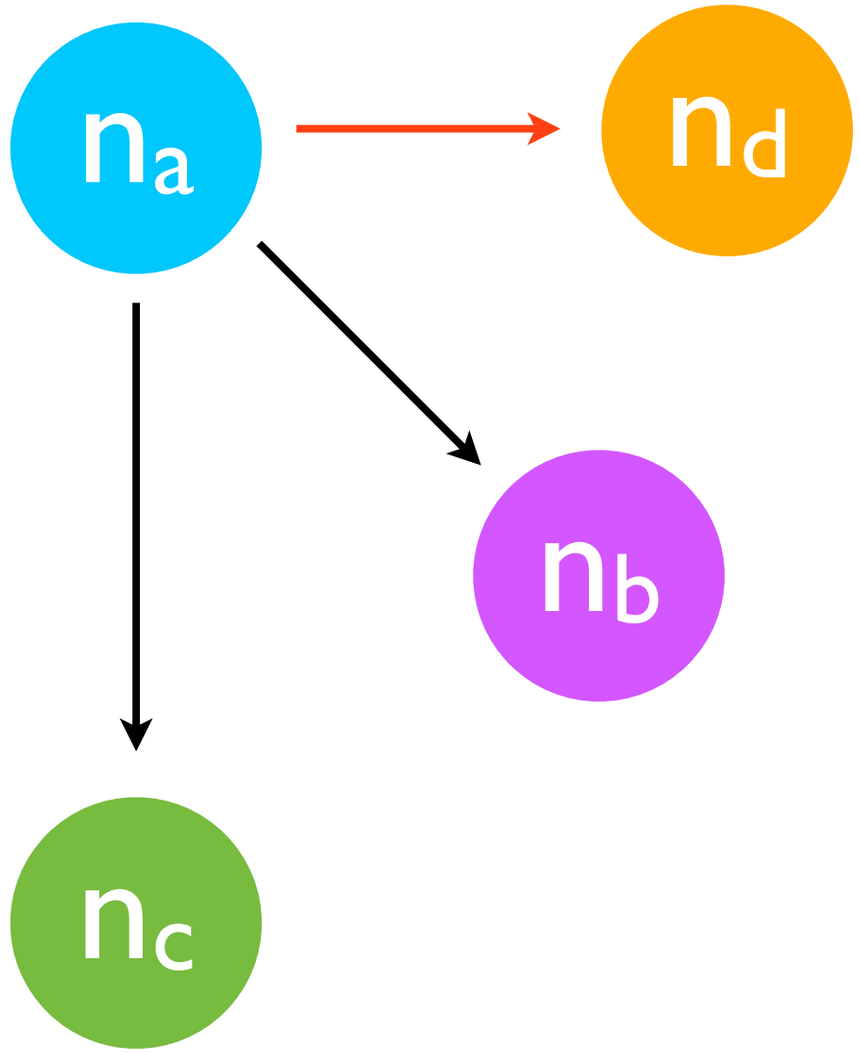
Dependency graphs

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.

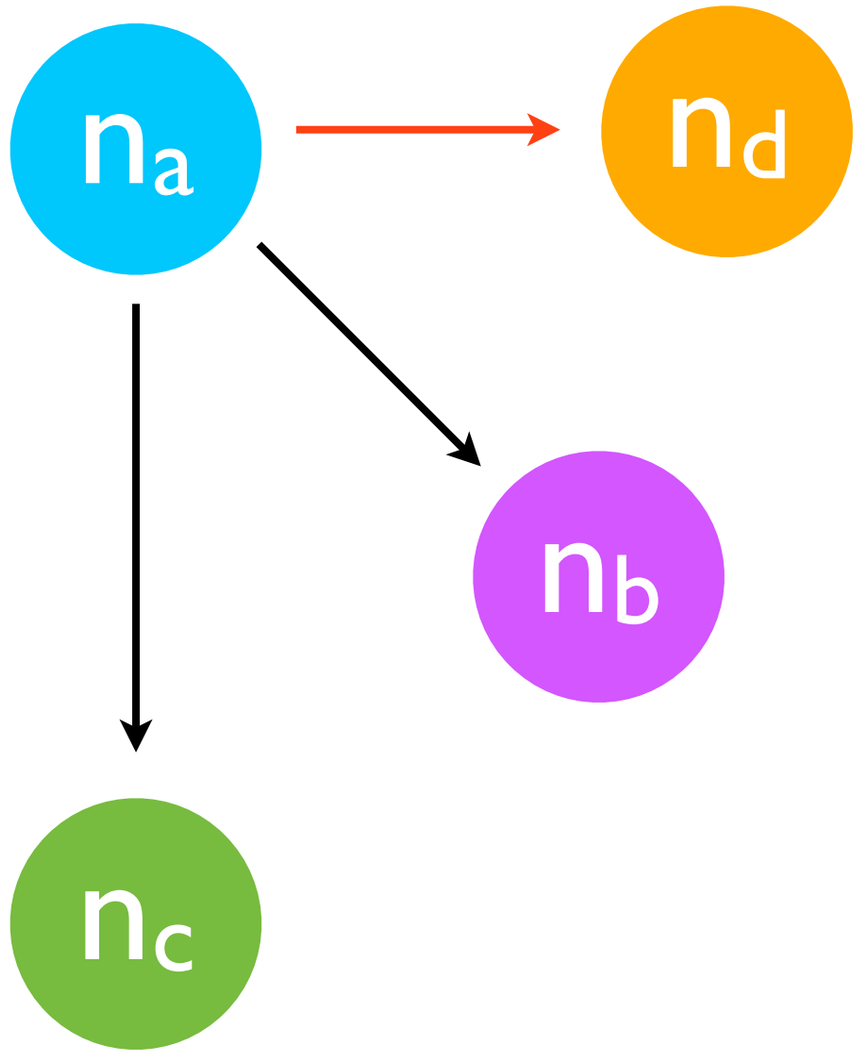
- Edges are colored by dependency type: **`dependencies`** or **`devDependencies`**



Dependency graphs

The G = (V,E) kind of graph

```
{
  "name": "package-a",
  "dependencies": {
    "package-b": "~1.0.4",
    "package-c": "~2.1.3"
  },
  "devDependencies": {
    "package-d": "~3.1.2"
  },
  "main": "./index.js"
}
```



Now imagine this **for 80,000+ packages!**


ANOTHER. MIND. EXPLOSION.

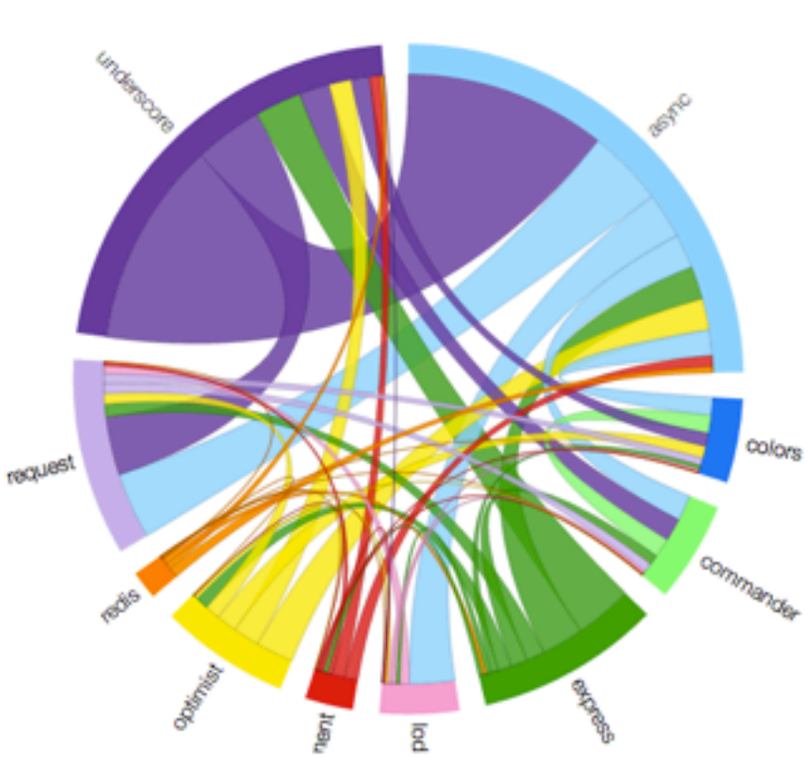
Actually.
It’s not so bad.

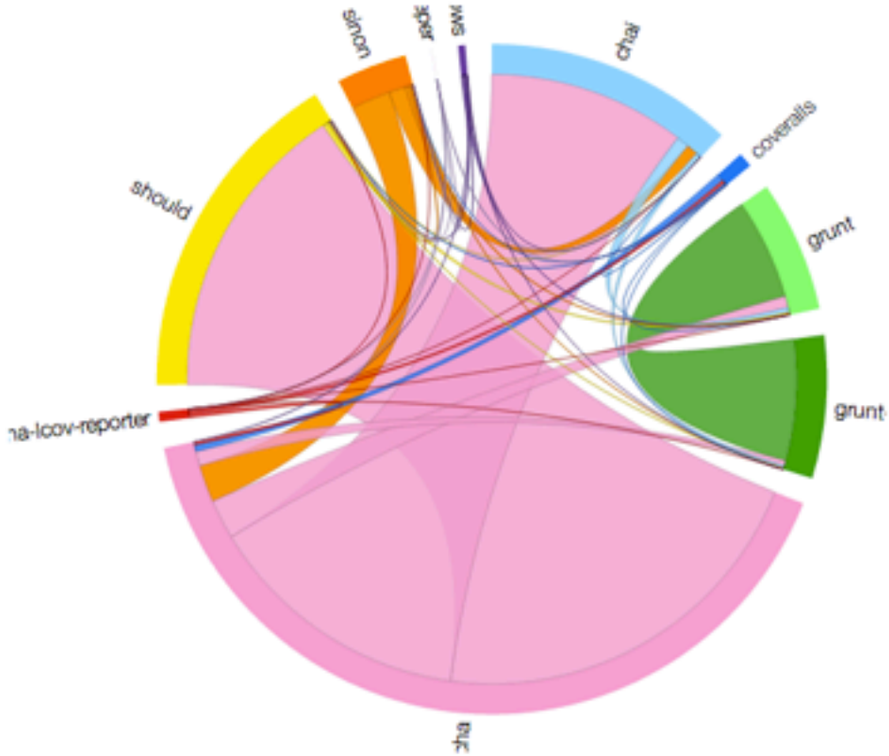
Dependency graphs

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 ...”*

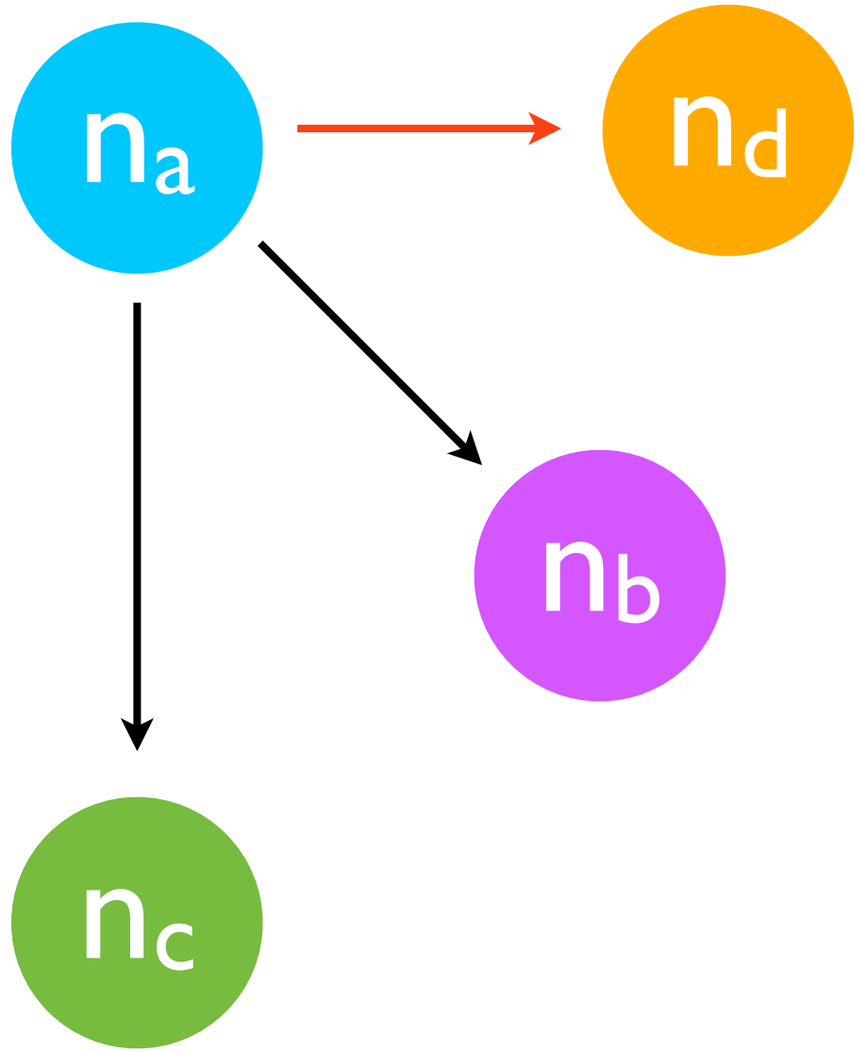
```
["package", "codep", "thru"]
```

<3 CouchDB!

Dependency graphs

The G = (V,E) kind of graph

```
{
  "name": "package-a",
  "dependencies": {
    "package-b": "~1.0.4",
    "package-c": "~2.1.3"
  },
  "devDependencies": {
    "package-d": "~3.1.2"
  },
  "main": "./index.js"
}
```



Here we would say that **package-b** and **package-c** have a **codependency relationship** thru **package-a**

Codependencies?

Thanks CouchDB!

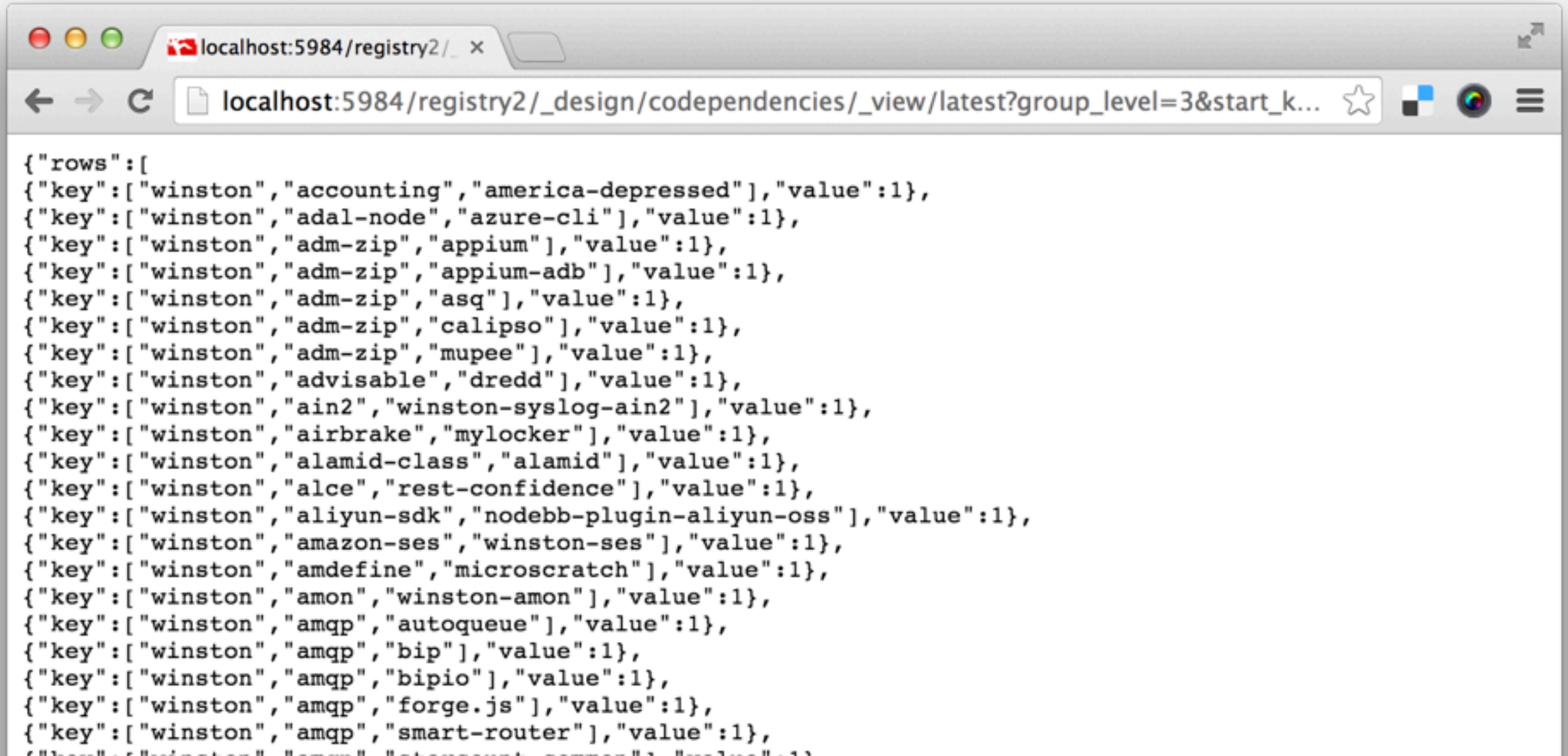
- We can get all of this from a **simple CouchDB view.**

```
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",{}]
```



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…	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

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 total
```

Codependencies

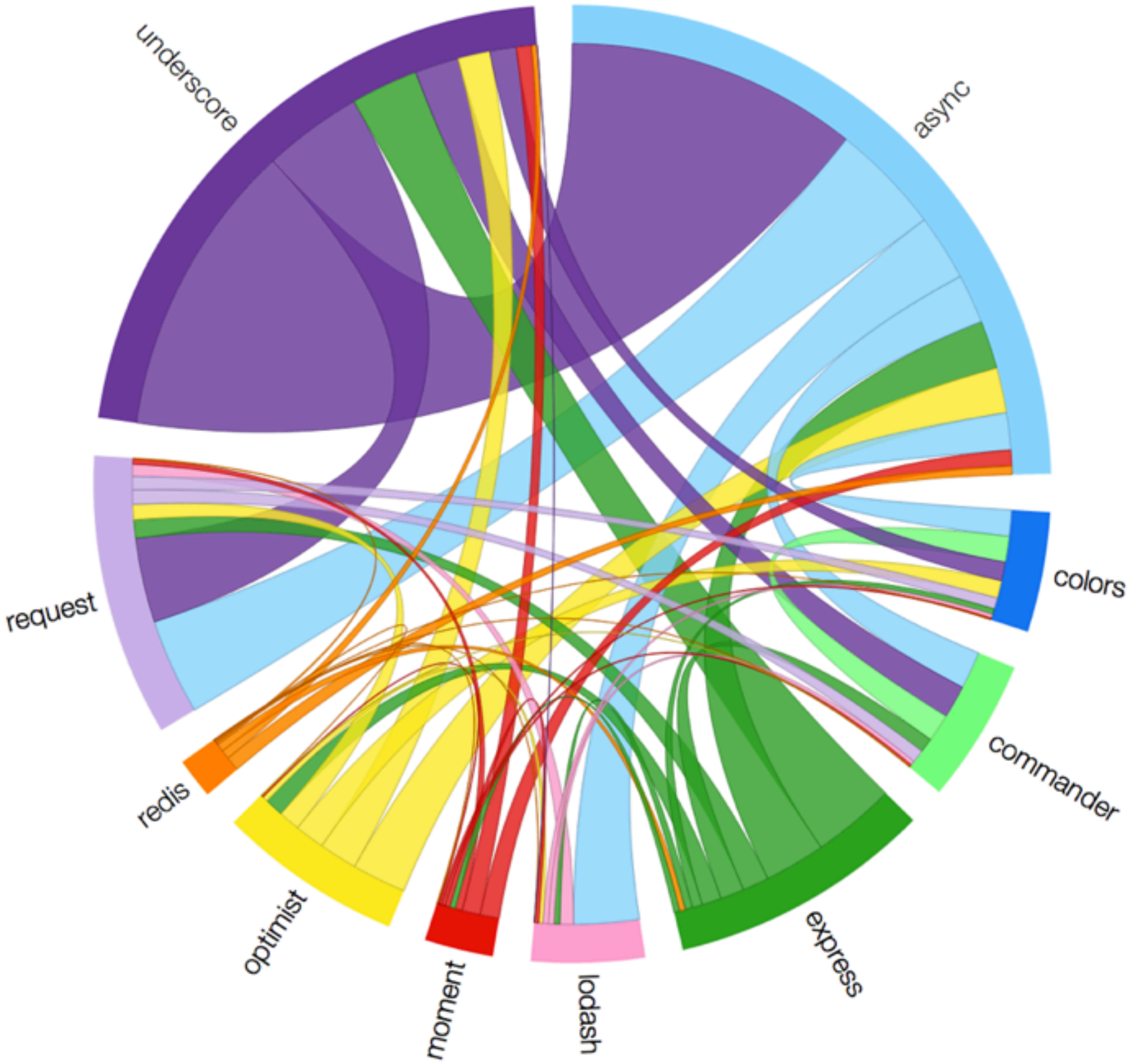
Understanding codependencies through winston

- To do this we go and calculate the codependencies for each of the members

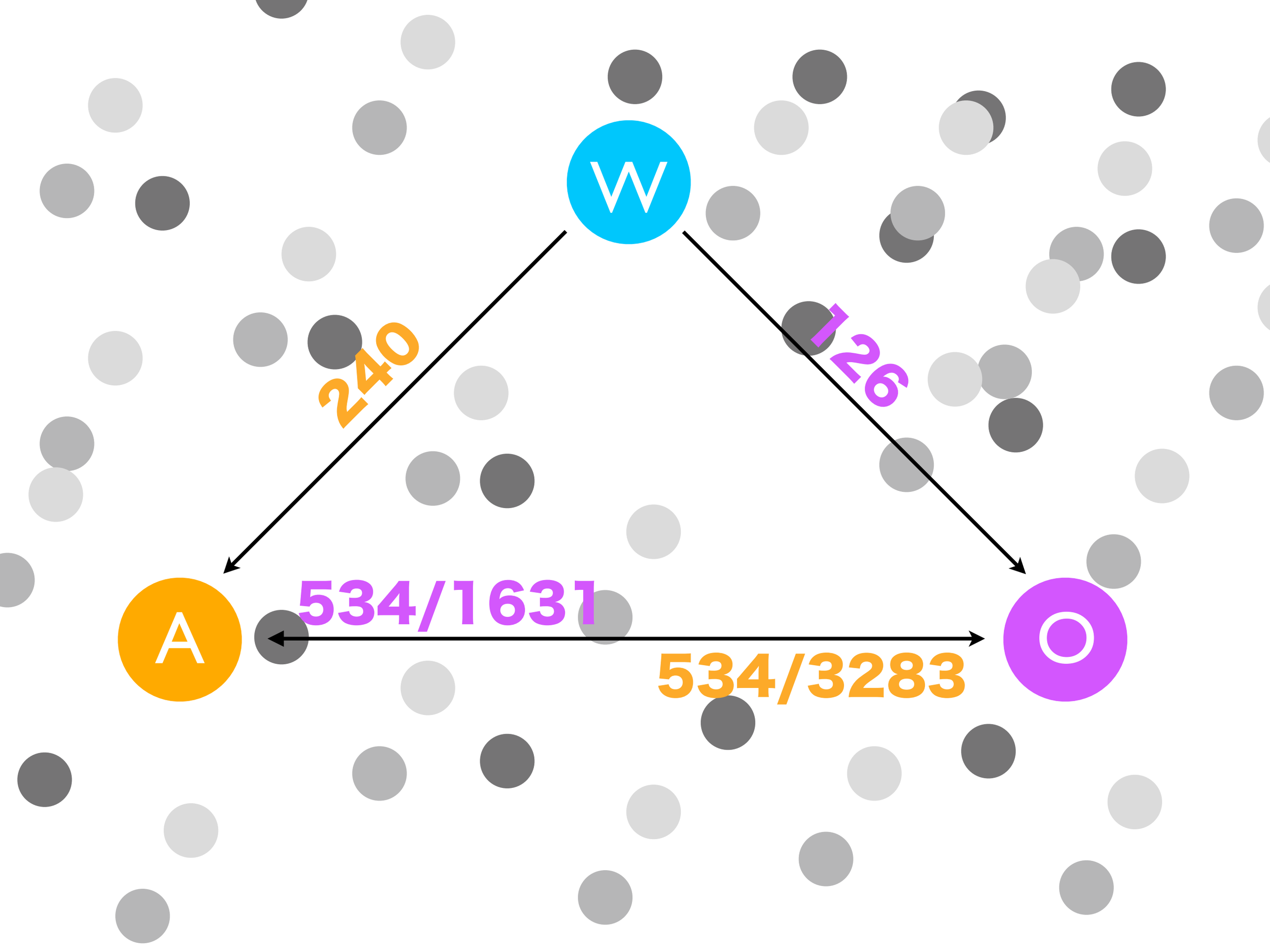
ASYNC

```
1359 | underscore| 0.4139
837  | request    | 0.2549
553  | express    | 0.1684
534  | optimist   | 0.1626
3283 total
```

AND THUS MEANINGFUL GRAPHS



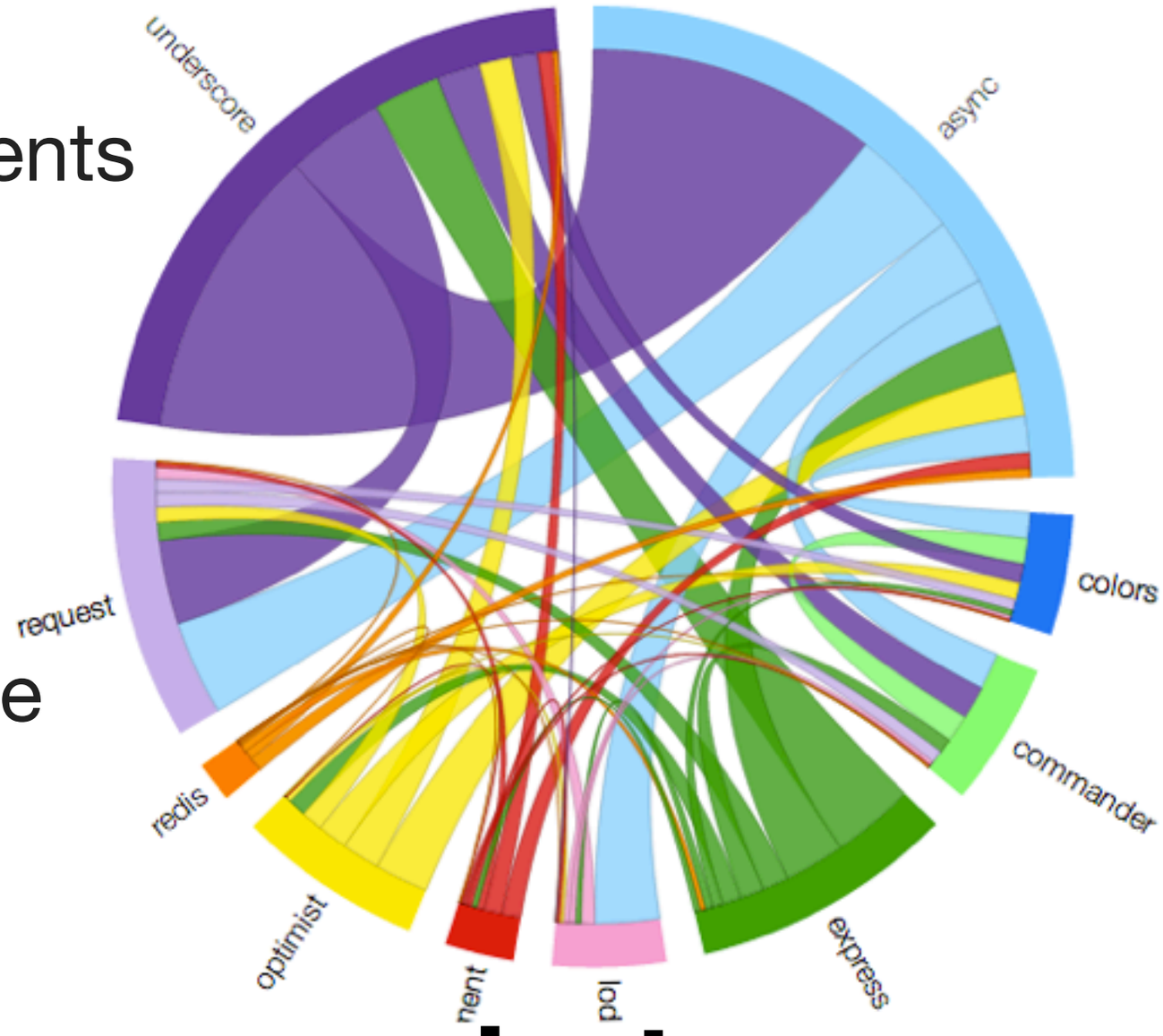
WAT?


W
240
126
A
534/1631
534/3283
O

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.



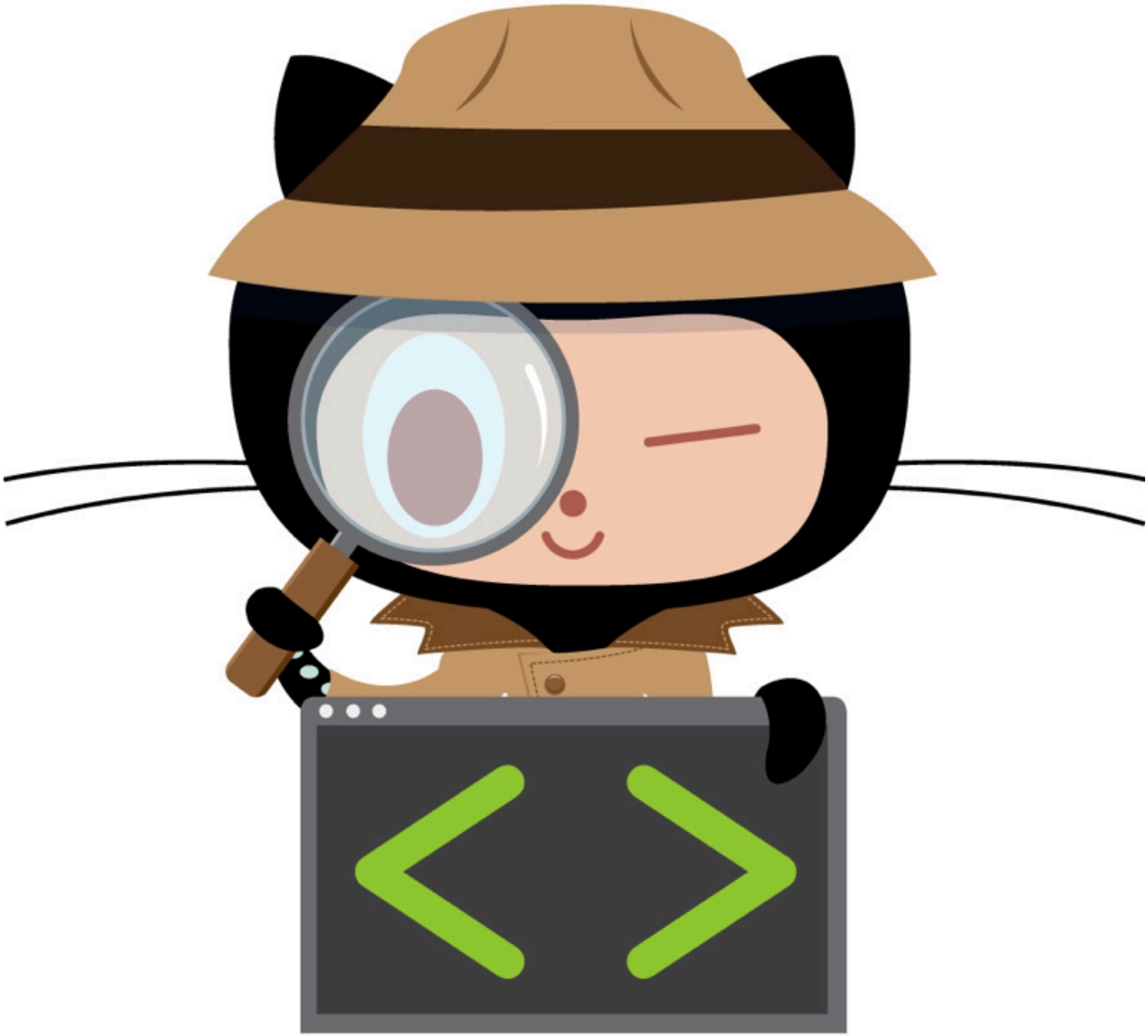
PAUSE FOR DEMO

ALL OPEN SOURCE

Seriously... what else do people do?

indexzero/npm-codependencies

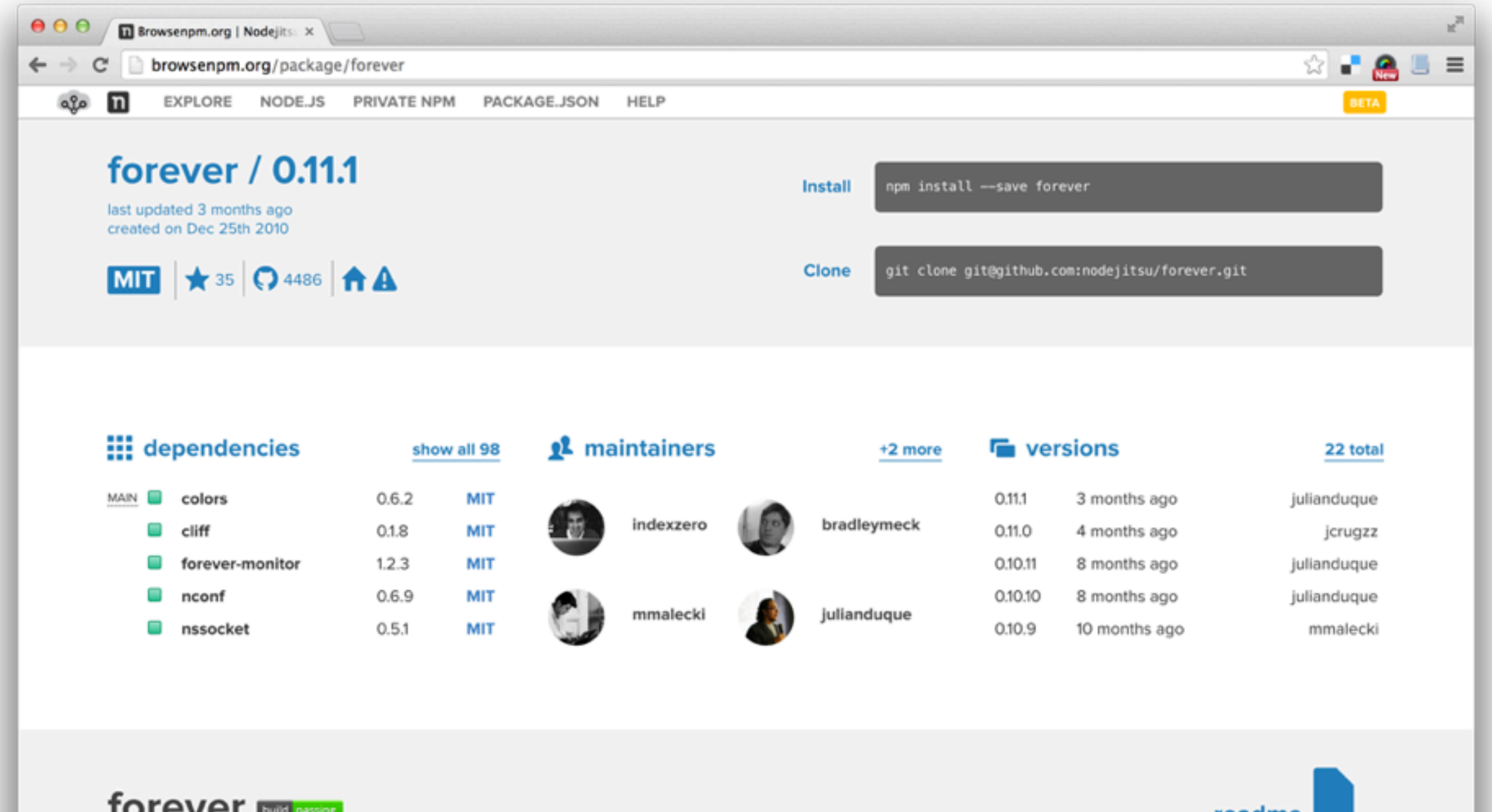
indexzero/npm-comp-stat-www




github

browsenpm.org

github.com/nodejitsu/browsenpm.org



BUT! THAT'S NOT ALL!

NO, NOT TOP HATS AND MONACLES

I should use static analysis with npm

ANALYZE ALL THE THINGS!

WAS THE 37th PRESIDENT

RICHARD NIXON

WAS A PRETTY AWFUL PRESIDENT

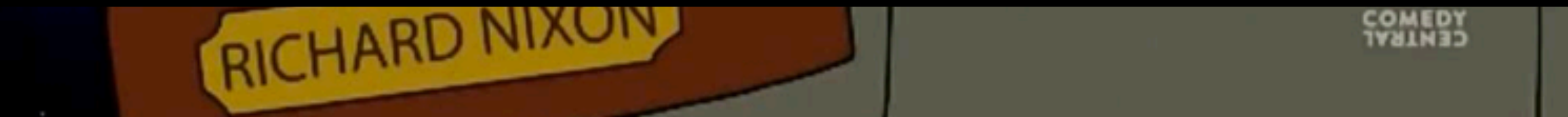
RICHARD NIXON

ISN'T REALLY THE POINT

RICHARD NIXON

GRATUITOUS HEAD IN A JAR

RICHARD NIXON



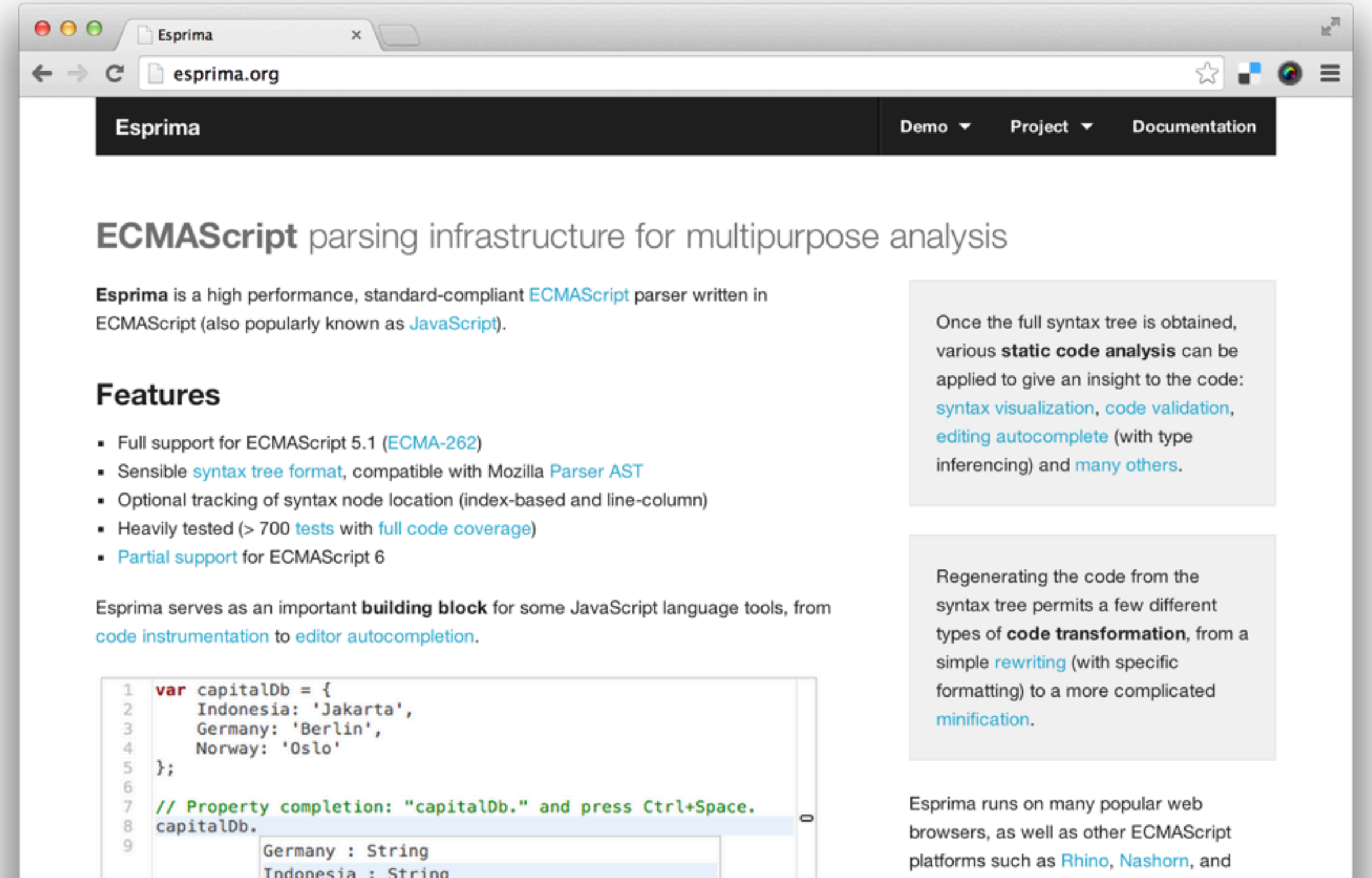
WELL THE FEAR OF A

SILENT MAJORITY

ANYWAY.....

I should use static analysis with npm

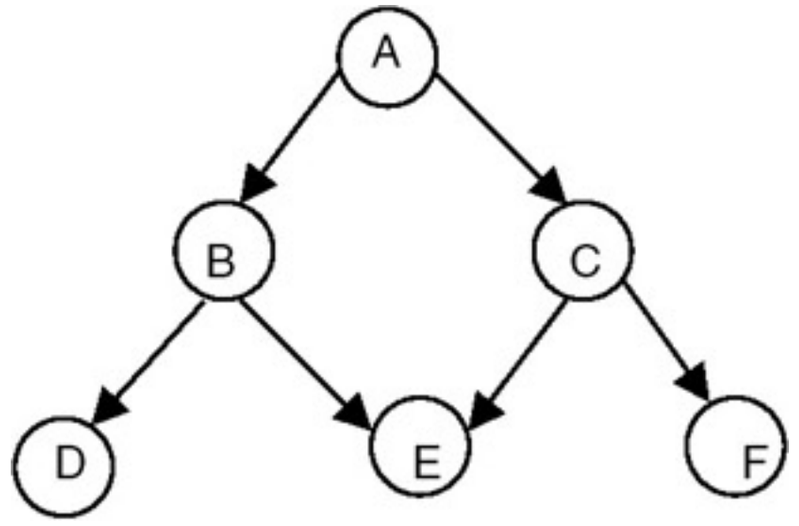
How the what exactly?



I should use static analysis with npm

How do you make the sausage?

ESPRIMA



CODE AST

NOW DO THAT FOR

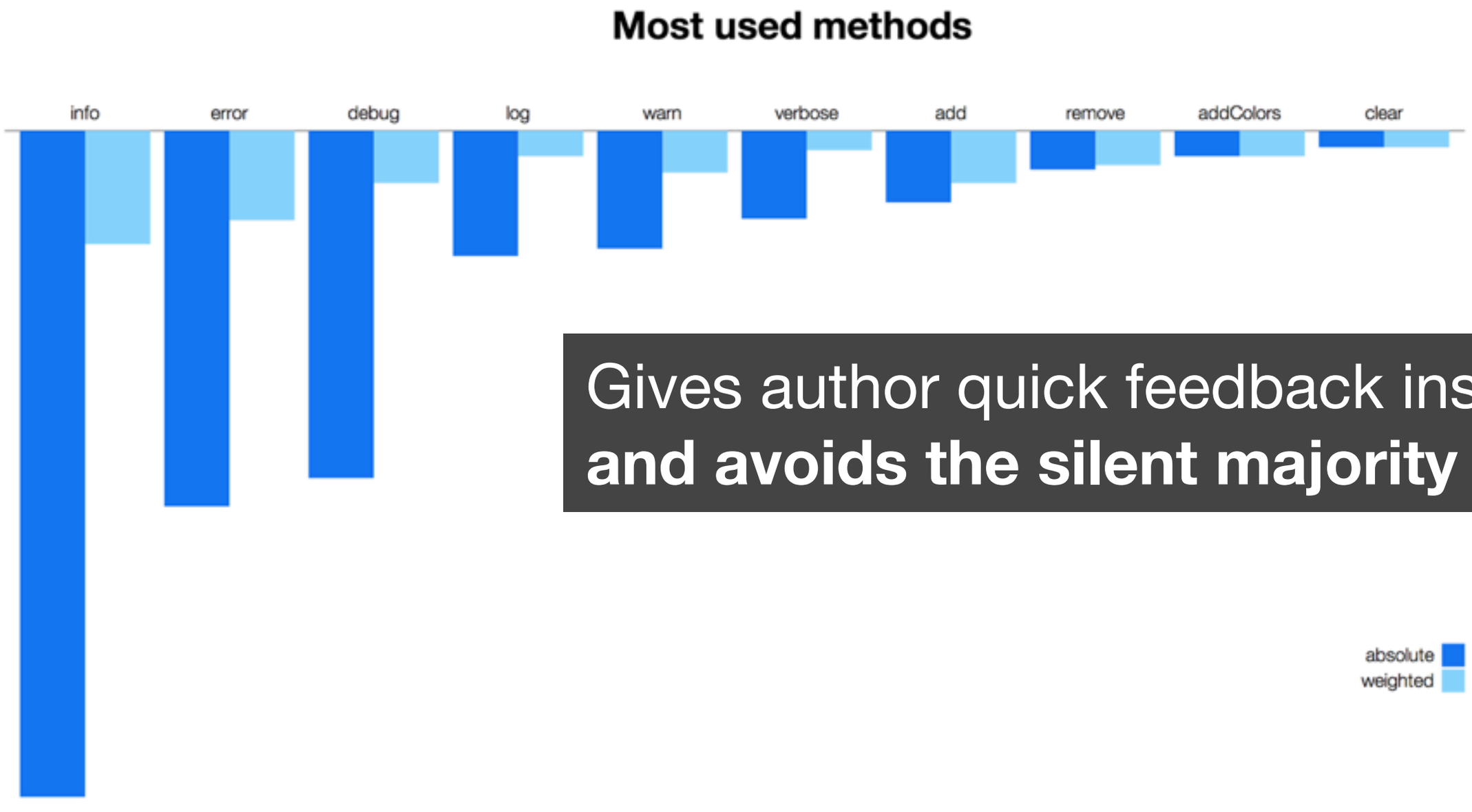
EVERY SOURCE FILE IN

EVERY DEPENDENT MODULE

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

ALL OPEN SOURCE

Yup... just like before.

indexzero/npm-static-stats

indexzero/npm-pipeline

nodejitsu

FIN