GEORGE MASON UNIVERSITY®

SWE 621

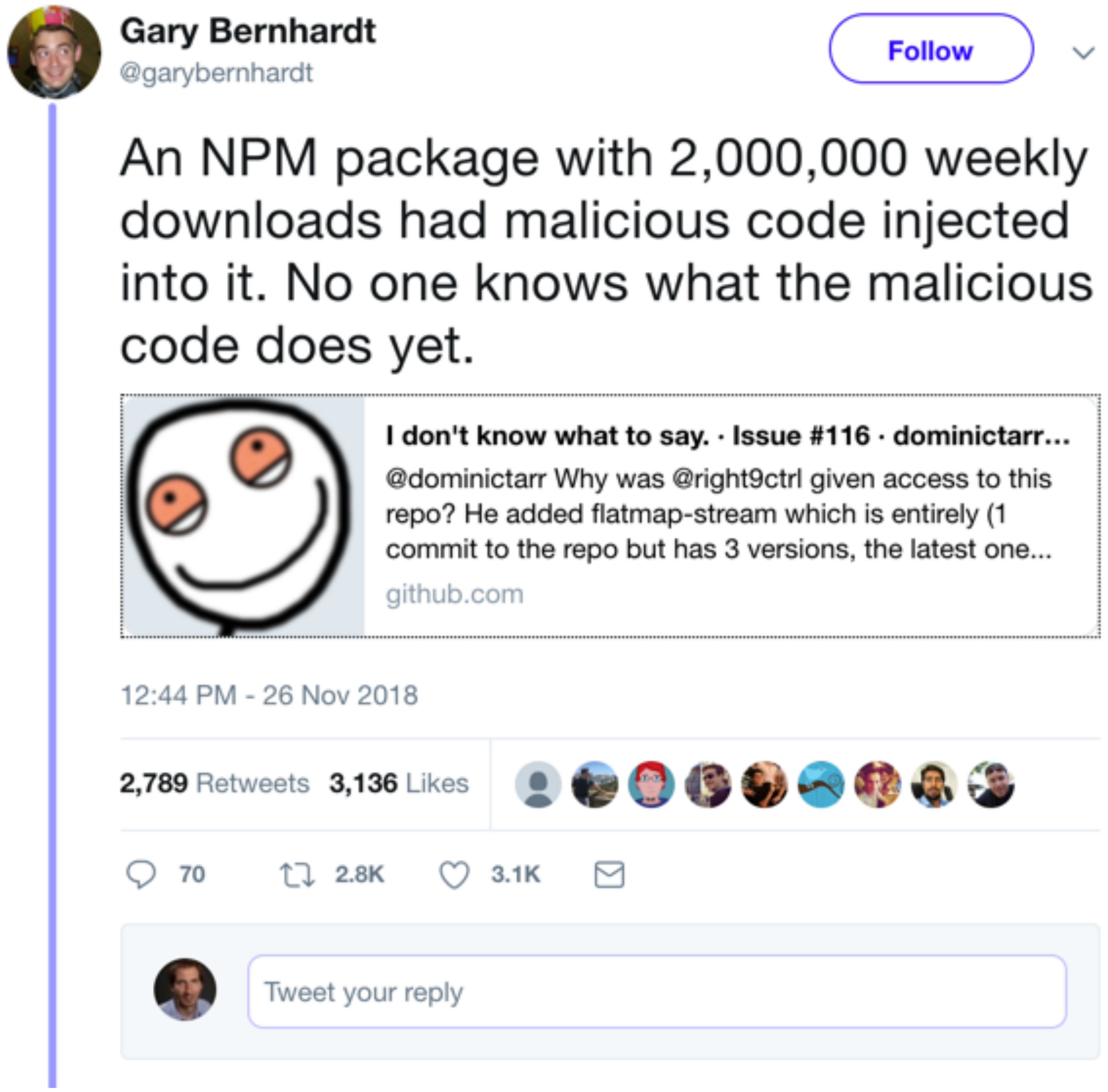
SPRING 2025

DESIGN ECOSYSTEMS

LOGISTICS

- HW5 due today
- Final Project Presentation: 5/5
- Review for final on 5/5
- Final Exam in two weeks on 5/12

EXAMPLE: NPM



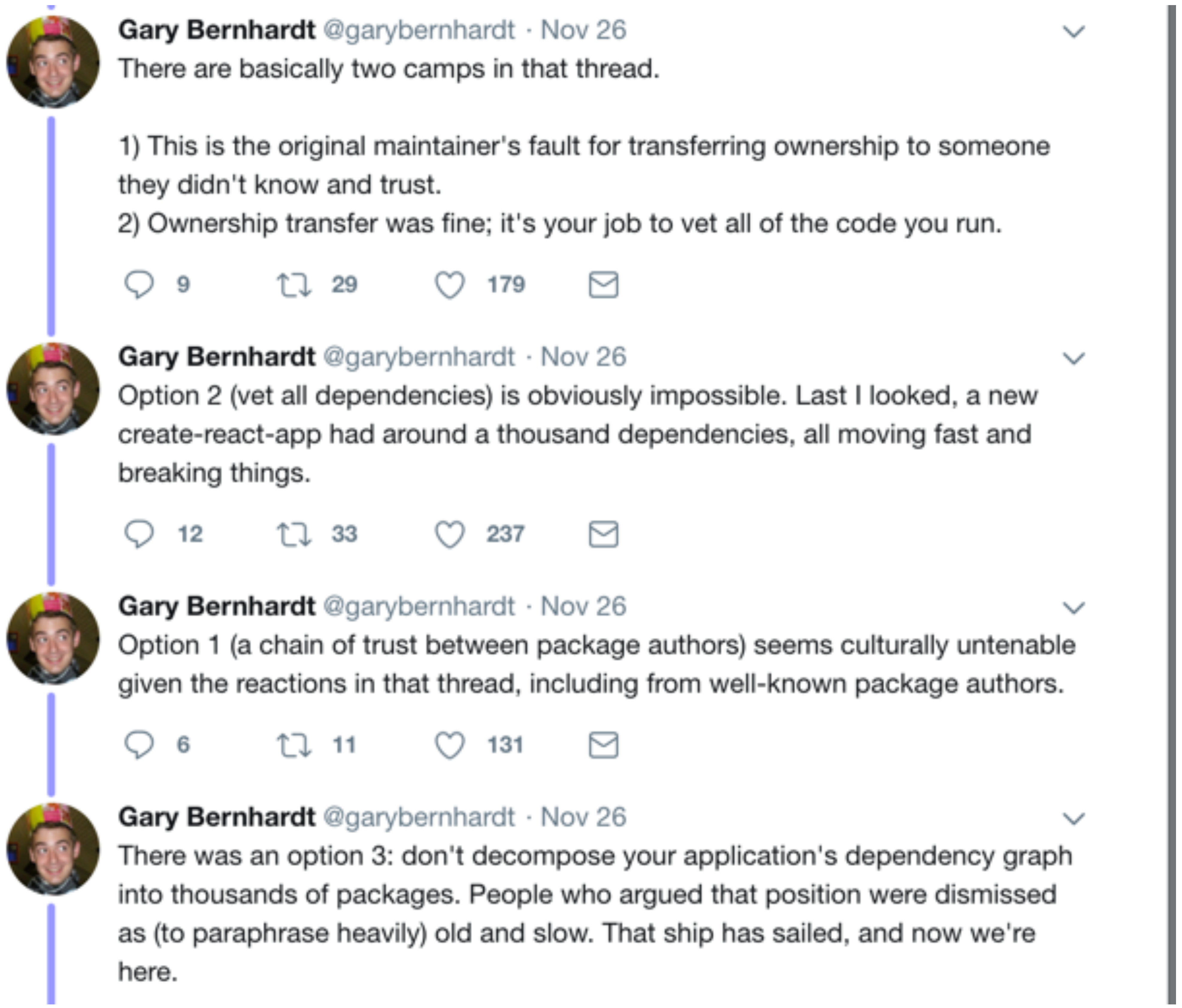
Gary Bernhardt
@garybernhardt

An NPM package with 2,000,000 weekly downloads had malicious code injected into it. No one knows what the malicious code does yet.

I don't know what to say. · Issue #116 · dominictarr...
@dominictarr Why was @right9ctrl given access to this repo? He added flatmap-stream which is entirely (1 commit to the repo but has 3 versions, the latest one...
github.com

12:44 PM - 26 Nov 2018

2,789 Retweets **3,136** Likes



Gary Bernhardt @garybernhardt · Nov 26
There are basically two camps in that thread.

1) This is the original maintainer's fault for transferring ownership to someone they didn't know and trust.
2) Ownership transfer was fine; it's your job to vet all of the code you run.

Gary Bernhardt @garybernhardt · Nov 26
Option 2 (vet all dependencies) is obviously impossible. Last I looked, a new create-react-app had around a thousand dependencies, all moving fast and breaking things.

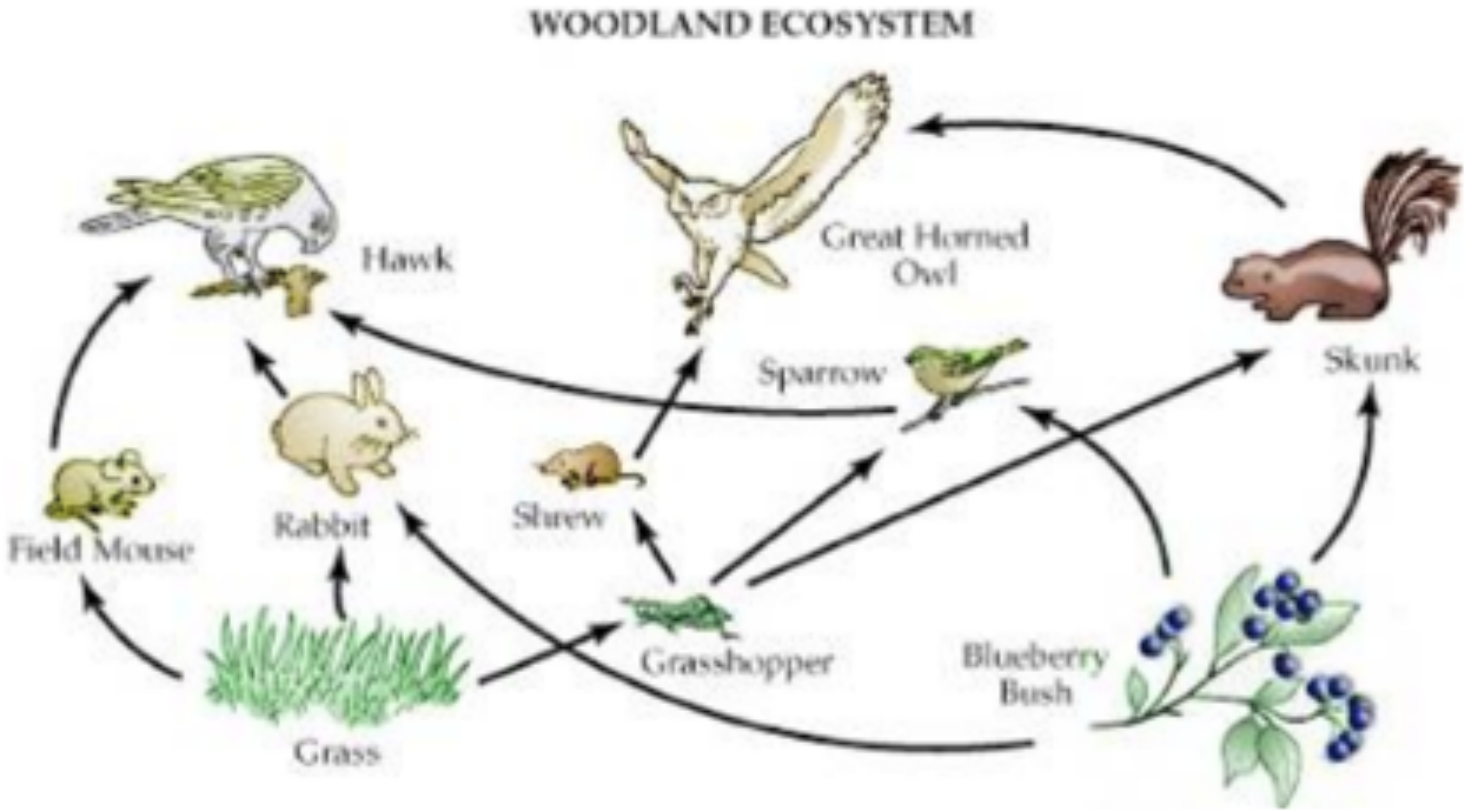
Gary Bernhardt @garybernhardt · Nov 26
Option 1 (a chain of trust between package authors) seems culturally untenable given the reactions in that thread, including from well-known package authors.

Gary Bernhardt @garybernhardt · Nov 26
There was an option 3: don't decompose your application's dependency graph into thousands of packages. People who argued that position were dismissed as (to paraphrase heavily) old and slow. That ship has sailed, and now we're here.

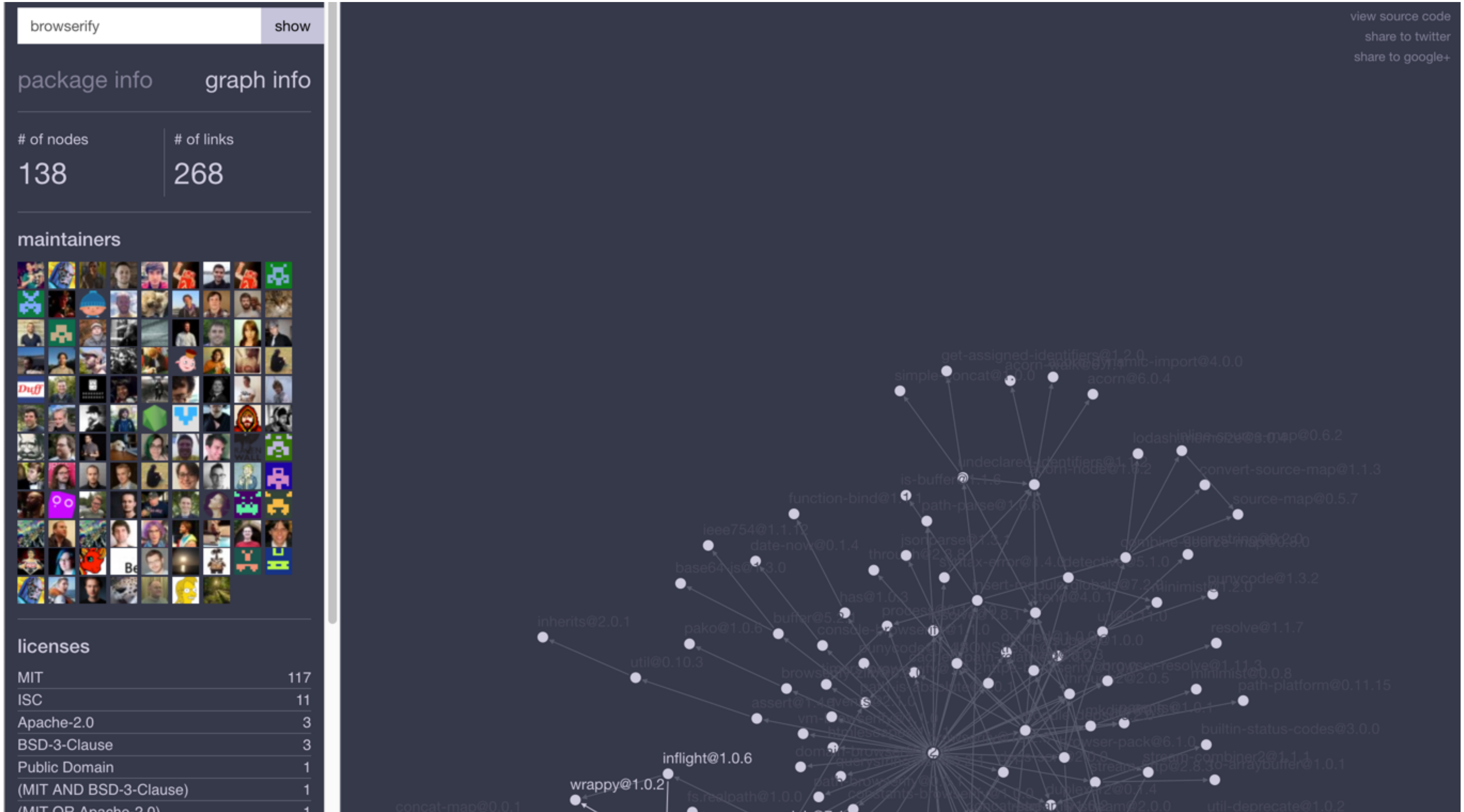
https://twitter.com/garybernhardt/status/1067111872225136640

ECOSYSTEMS

- Networks of systems with complex dependencies
  - Each system may have a separate organization / individual responsible for its creation and maintenan
- Key characteristic: **no single point of control**
  - Multiple individuals / organizations with separate goals, needs
  - But have deep interdependencies, where organizations depend on others in order to exist
  - Fundamentally differs from traditional design proces where is no single organization that makes design decisions that impact final system
- Dependencies both social and technical
  - Technical: our code uses your code
  - Social: we'd really like to have some input on what your code does and output on how it does it
- Driven by quest for **scale**



SOFTWARE ECOSYSTEMS EXAMPLE: NPM PACKAGES



browserify show

package info graph info

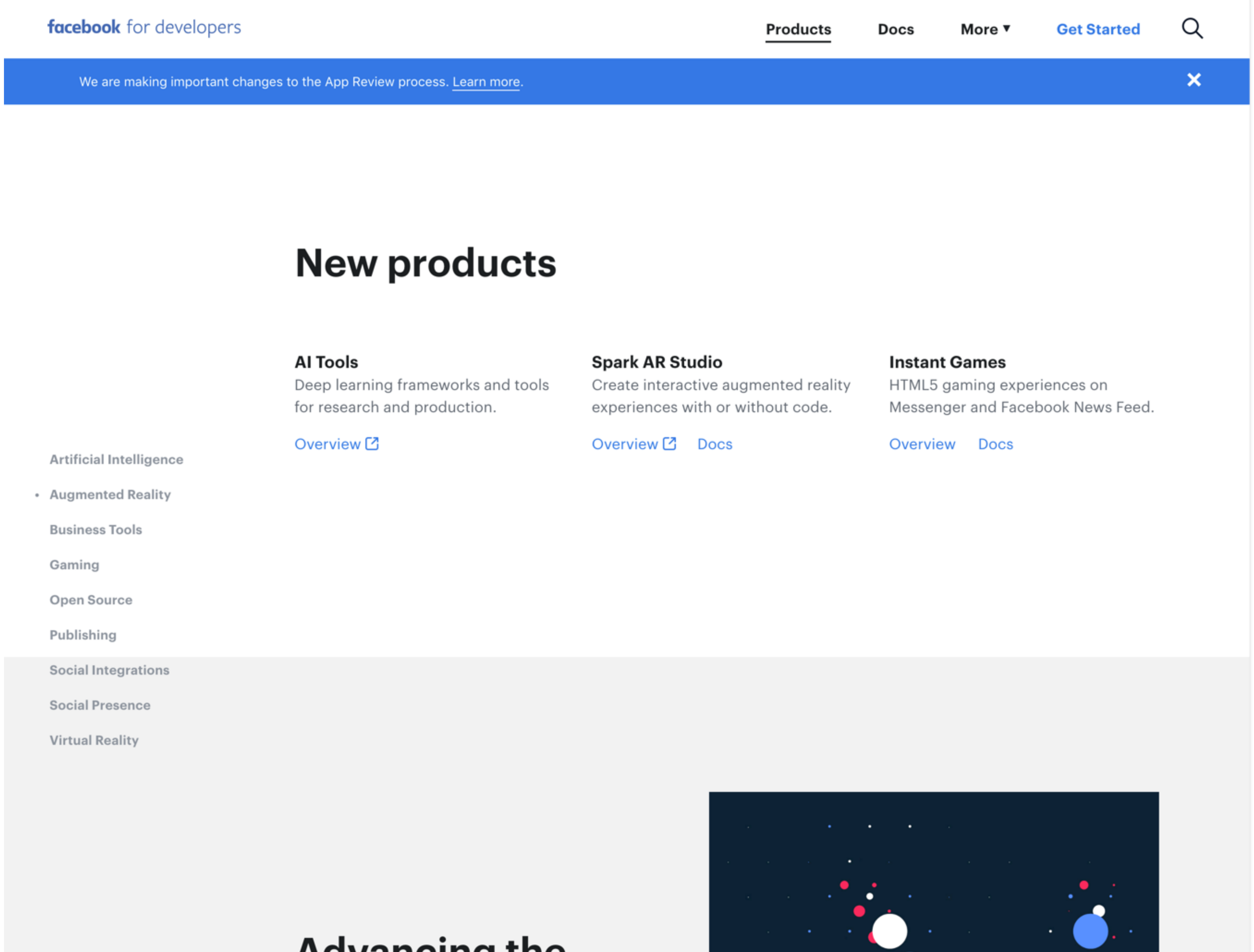
# of nodes	# of links
138	268

maintainers

licenses

MIT	117
ISC	11
Apache-2.0	3
BSD-3-Clause	3
Public Domain	1
(MIT AND BSD-3-Clause)	1
(MIT OR Apache-2.0)	1

SOFTWARE ECOSYSTEMS: FACEBOOK



facebook for developers

Products Docs More ▾ Get Started

We are making important changes to the App Review process. Learn more. ✕

New products

AI Tools
Deep learning frameworks and tools for research and production.
Overview

Spark AR Studio
Create interactive augmented reality experiences with or without code.
Overview Docs

Instant Games
HTML5 gaming experiences on Messenger and Facebook News Feed.
Overview Docs

- Artificial Intelligence
- Augmented Reality
- Business Tools
- Gaming
- Open Source
- Publishing
- Social Integrations
- Social Presence
- Virtual Reality

Advancing the

ACTIVITY

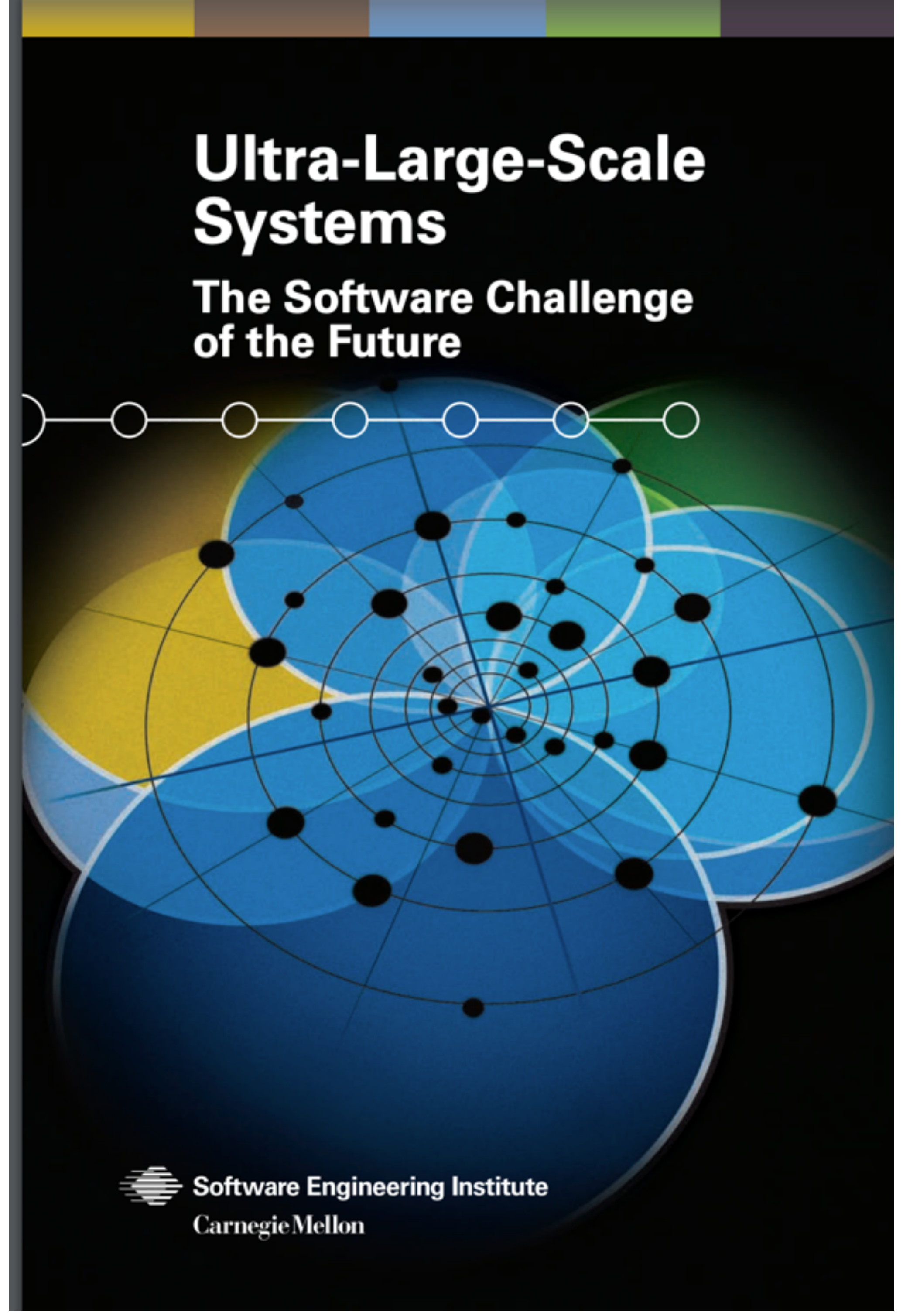
- What are other examples of software ecosystems?

SOME CHARACTERISTICS OF SOFTWARE ECOSYSTEMS

- Increase value of the core offering to existing users
- Increase attractiveness for new users
- Increase “stickiness” of the application platform
- Accelerate innovation through open innovation in the ecosystem
- Collaborate with partners in the ecosystems to share cost of innovation
- Platformize functionality developed by partners in the ecosystem (once success has been proven)

FIRST CHARACTERIZED IN CONTEXT OF MILITARY SYSTEMS

- "Ultra large scale systems"
  - Book published by Software Engineering Institute in 2006
- Large scale in terms of number of people, amount of data, number of interdependencies
- Decentralized in a variety of ways
- Developed and used by a wide variety of stakeholders with conflicting needs
- Constructed from heterogeneous parts.
- Software and hardware failures will be the norm rather than the exception.
- More like a **city** than a building


Ultra-Large-Scale Systems
The Software Challenge of the Future
Software Engineering Institute
Carnegie Mellon

https://resources.sei.cmu.edu/asset_files/Book/2006_014_001_30542.pdf

TYPES OF ECOSYSTEMS

- Can describe ecosystems by the key player in the ecosystem
- Usually the organization that owns the key API
    - Power accrues to organization by controlling what the API can and cannot do
    - But organization needs API users to be successful
- In some cases, may be no key player (e.g., NPM package ecosystem)

OS-CENTRIC ECOSYSTEMS

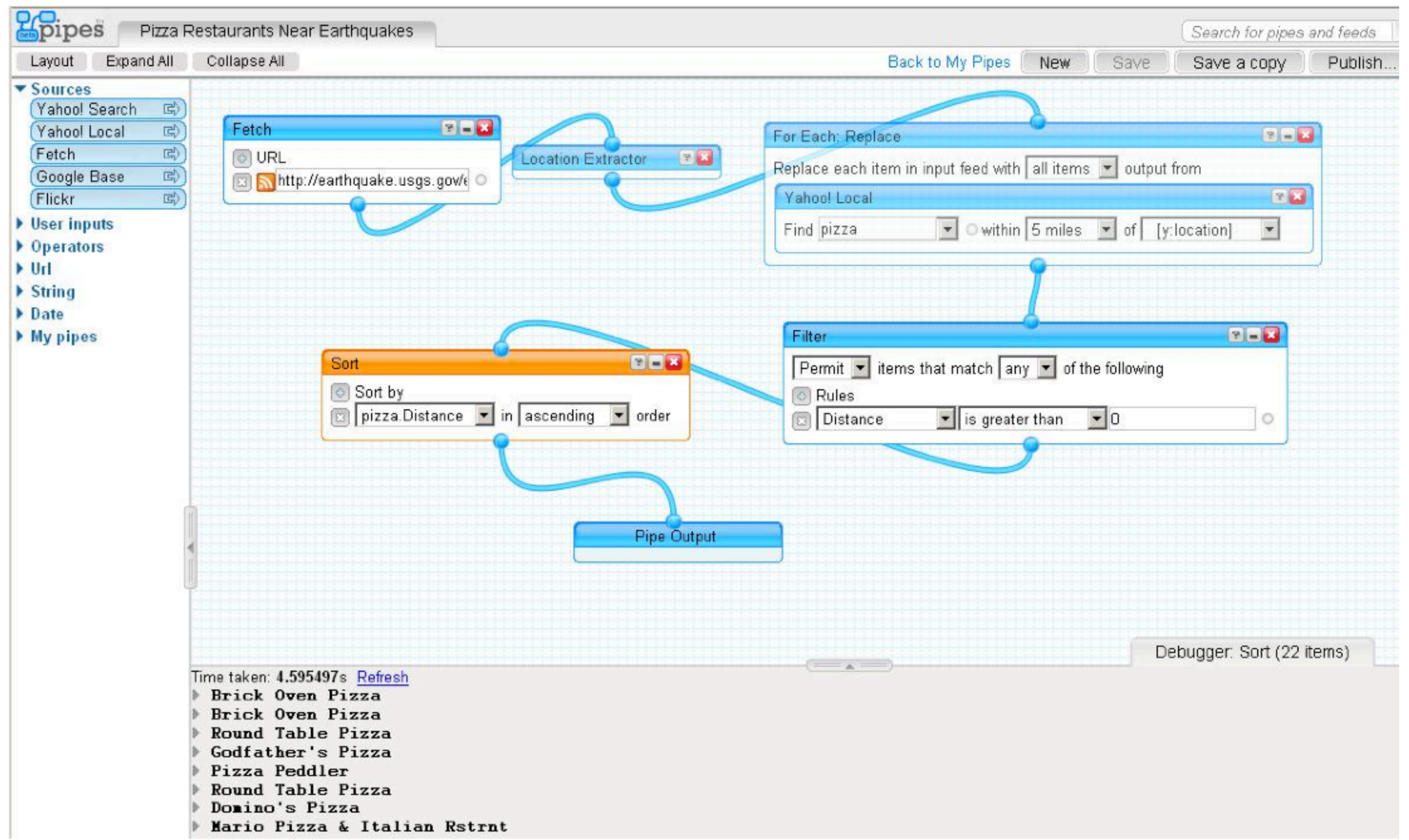
- Linux vs. Windows vs. OS X
- node.js vs. Java servlets vs. PHP
- Platform which offers an API which others build on top of
- Platform abstracts over complexity of underlying hardware
- Key player goal: increase value of platform by getting more users

APPLICATION-CENTRIC ECOSYSTEMS

- Facebook vs. Twitter vs. LinkedIn
- Word vs. Google Docs
- User facing application which exposes points where 3rd party developers can extend application
- Key player goal: increase value of application by getting more developers to build more functionality

END-USER PROGRAMMING ECOSYSTEMS

- Microsoft Excel vs. Yahoo Pipes vs. Scratch
- Domain specific language (DSL) offers a simpler way to program for those who are not professional software developers (e.g., kids, scientists, financial analysts)
- Platform offers language, programming environment, and (sometimes) repository of programs which can be remixed



URL: http://earthquake.usgs.gov/e

Replace each item in input feed with all items output from

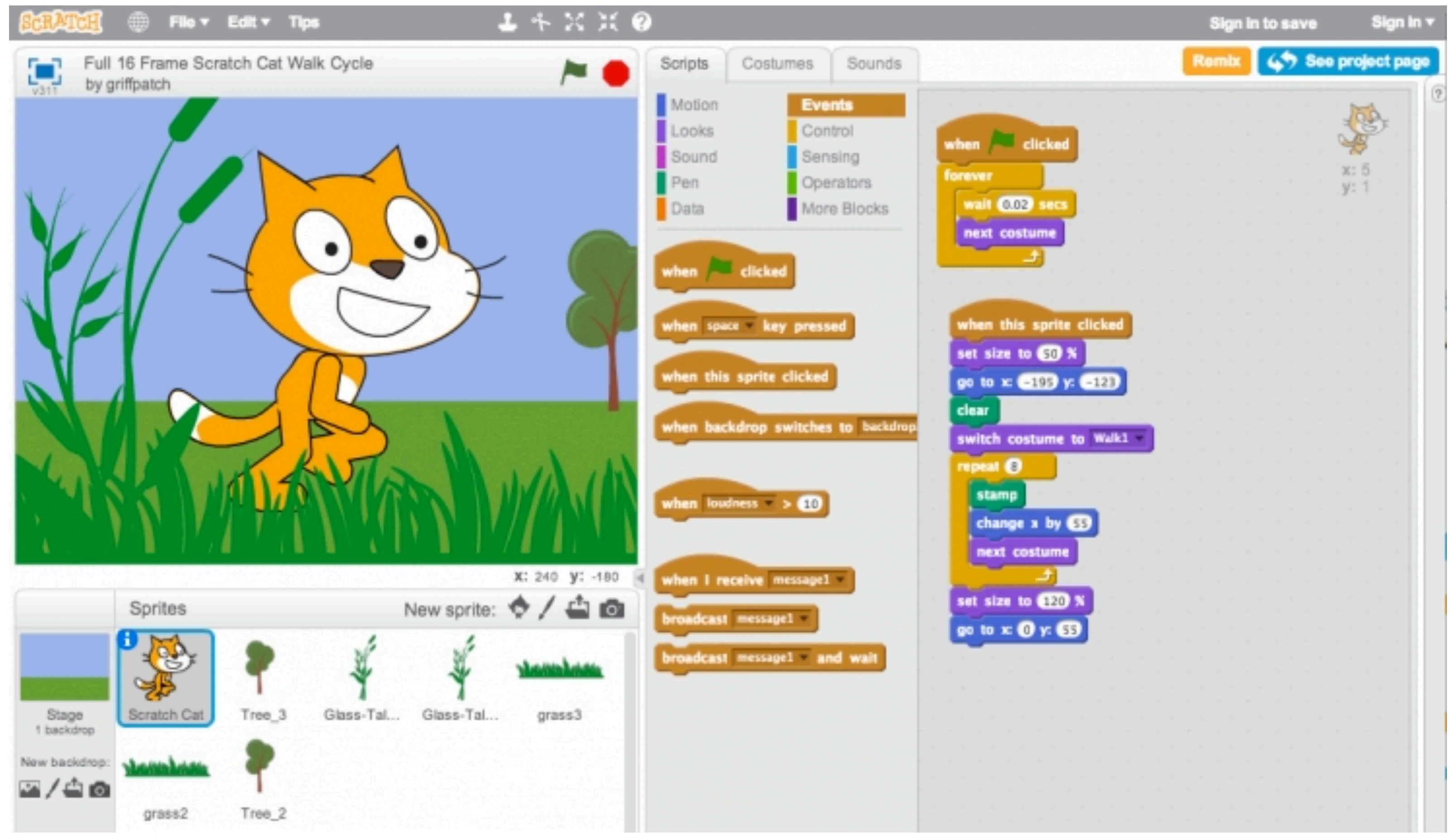
Find pizza within 5 miles of [y:location]

Permit items that match any of the following

Rules: Distance is greater than 0

Sort by pizza.Distance in ascending order

```
Time taken: 4.595497s Refresh
▸ Brick Oven Pizza
▸ Brick Oven Pizza
▸ Round Table Pizza
▸ Godfather's Pizza
▸ Pizza Peddler
▸ Round Table Pizza
▸ Domino's Pizza
▸ Mario Pizza & Italian Rstrnt
```



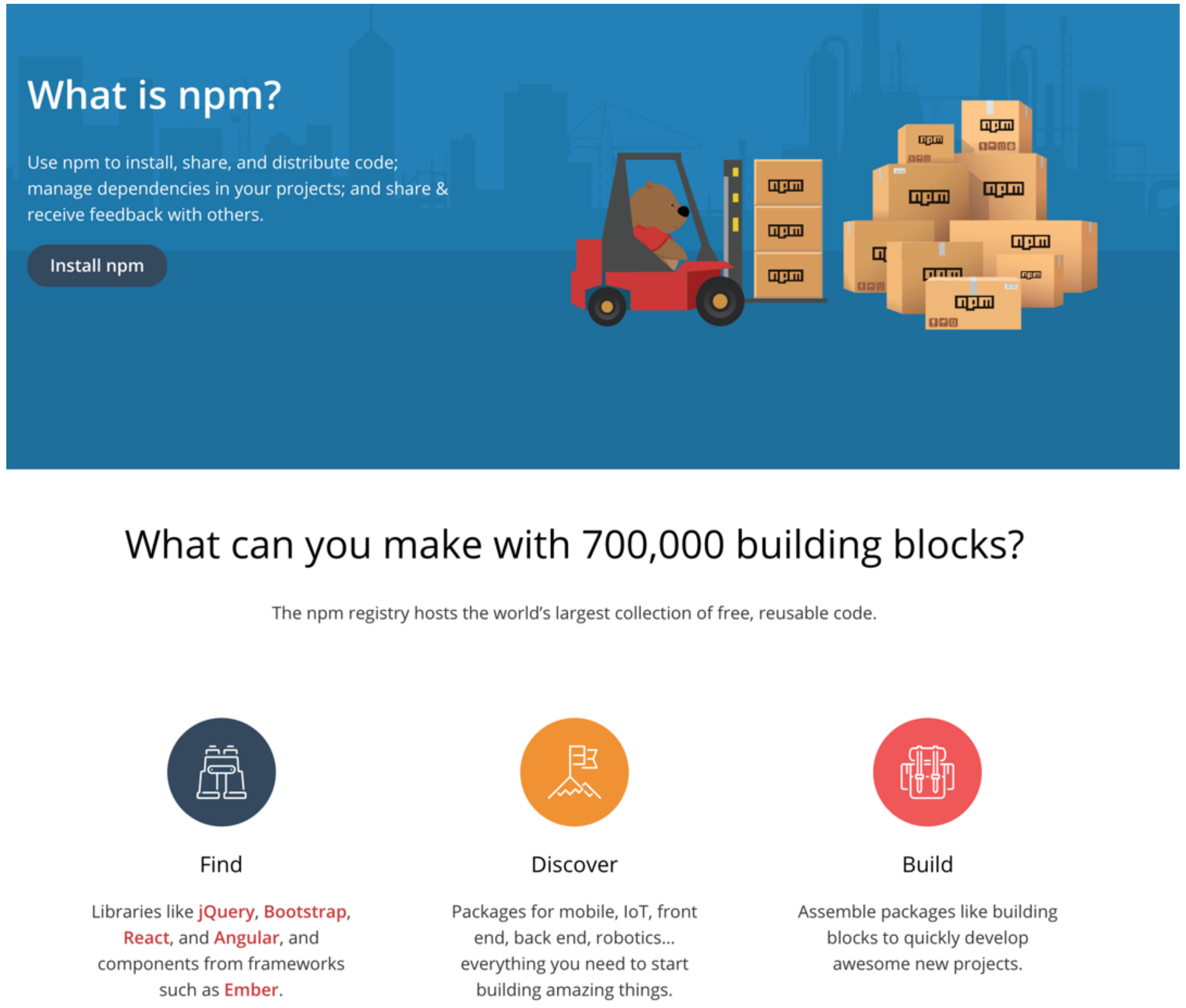
```
when clicked
forever
    wait 0.02 secs
    next costume

when this sprite clicked
set size to 50 %
go to x: -195 y: -123
clear
switch costume to Walk1
repeat 8
    stamp
    change x by 55
    next costume
set size to 120 %
go to x: 0 y: 55
```

PACKAGE-CENTRIC ECOSYSTEMS

- NPM vs. R vs. Ruby
- Individual publish **packages** in central repository
- Packages may depend on other packages
- Automated build process automatically fetches package from repository, enabling automatic updating to latest version
- Popular packages gain recognition, encourage reuse by others

EXAMPLE: NODE PACKAGE MANAGER (NPM) ECOSYSTEM

- Node.js is runtime environment for server-side JavaScript applications
- Node package manager is an online repository of packages containing over 700,000 packages
- Core value: make it easy to publish, use, and **rapidly change** packages
  - Resulted in large repository of packages that are very widely used in web applications



What is npm?

Use npm to install, share, and distribute code; manage dependencies in your projects; and share & receive feedback with others.

What can you make with 700,000 building blocks?

The npm registry hosts the world's largest collection of free, reusable code.

Find

Libraries like jQuery, Bootstrap, React, and Angular, and components from frameworks such as Ember.

Discover

Packages for mobile, IoT, front end, back end, robotics... everything you need to start building amazing things.

Build

Assemble packages like building blocks to quickly develop awesome new projects.

DEVELOPERS, DEVELOPERS, DEVELOPERS

https://www.youtube.com/watch?v=Vhh_GeBPOhs

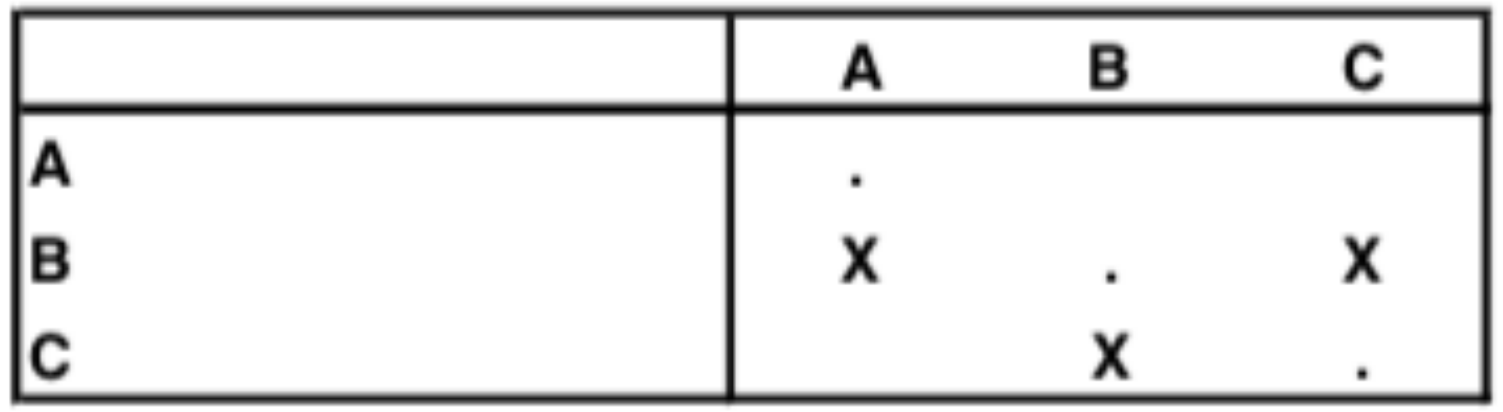
DEVELOPERS

- Key goal: increase value of ecosystem for owner by encouraging more people to use it
- Key player benefits from **scale**, which increases value of their platform
- Others may benefits as well
  - More scale --> more StackOverflow posts, tutorials, knowledgable developers --> easier to use

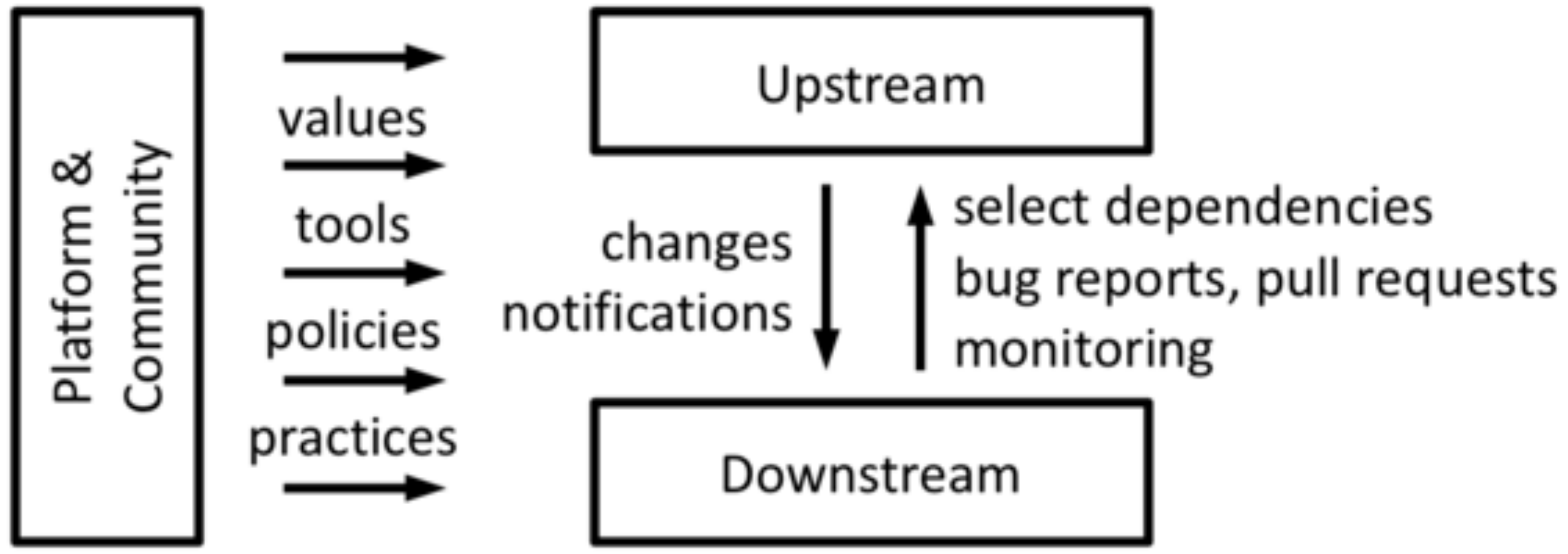
DEPENDENCIES

- Central to ecosystem is a **dependency,** where **downstream** system depends on **upstream** system

- Can think about individual dependencies (e.g., we depend on this method in this element) or just that there is a dependency (e.g., there exists some dependency from our project on project x)



	A	B	C
A	.		
B	X	.	X
C		X	.



Platform & Community

values

tools

policies

practices

Upstream

changes
notifications

select dependencies
bug reports, pull requests
monitoring

Downstream

CHANGE

- What happens when an upstream system introduces a change?
- Backwards compatible change: upstream system provides everything they did before and more
  - Nothing needs to change on downstream system
  - Just have new functionality to be used
- Breaking change: upstream system no longer fulfills contract it did before
  - Method might be deprecated, renamed, or changed in its behavior
- Burden of change
  - Downstream system will not work until is updated to work with new version

IN-CLASS ACTIVITY

- Have you needed to respond to a breaking change?
- Have you introduced a breaking change?

WHY DO BREAKING CHANGES HAPPEN?

- Imposes burden on downstream projects, so why would anyone do this?
- Technical debt: current version has poorly chosen object models or method names, lack of extensibility, little used methods
  - Determine better way of exposing functionality, introducing through backwards compatible change
  - Introduces cost to maintain old API
  - Old API adds confusion, where there's multiple ways of doing things that confuses new developers
- Efficiency: faster implementation requires new API
- Fixing defects: implementation incorrect, but downstream project relies on incorrect behavior
  - Downstream projects may have workaround for defect, which may break when defect is fixed

TECHNIQUES TO MITIGATE OR DELAY COSTS

- Maintain old interfaces
  - Deprecate interfaces but still keep supporting them
- Maintain multiple parallel releases
  - Multiple major versions with breaking changes
  - Keep incorporating minor changes (e.g,. security patches) for older versions
- Expose different APIs for different users
  - Detailed and frequently updated API for sophisticated users, higher level and more stable API for casual users
- Reduce number of releases with breaking changes
- Communicate breaking changes in advance
  - Include documentation and/or tool support for migrating clients to new versions

EXAMPLE: BREAKING CHANGES IN ECLIPSE ECOSYSTEM

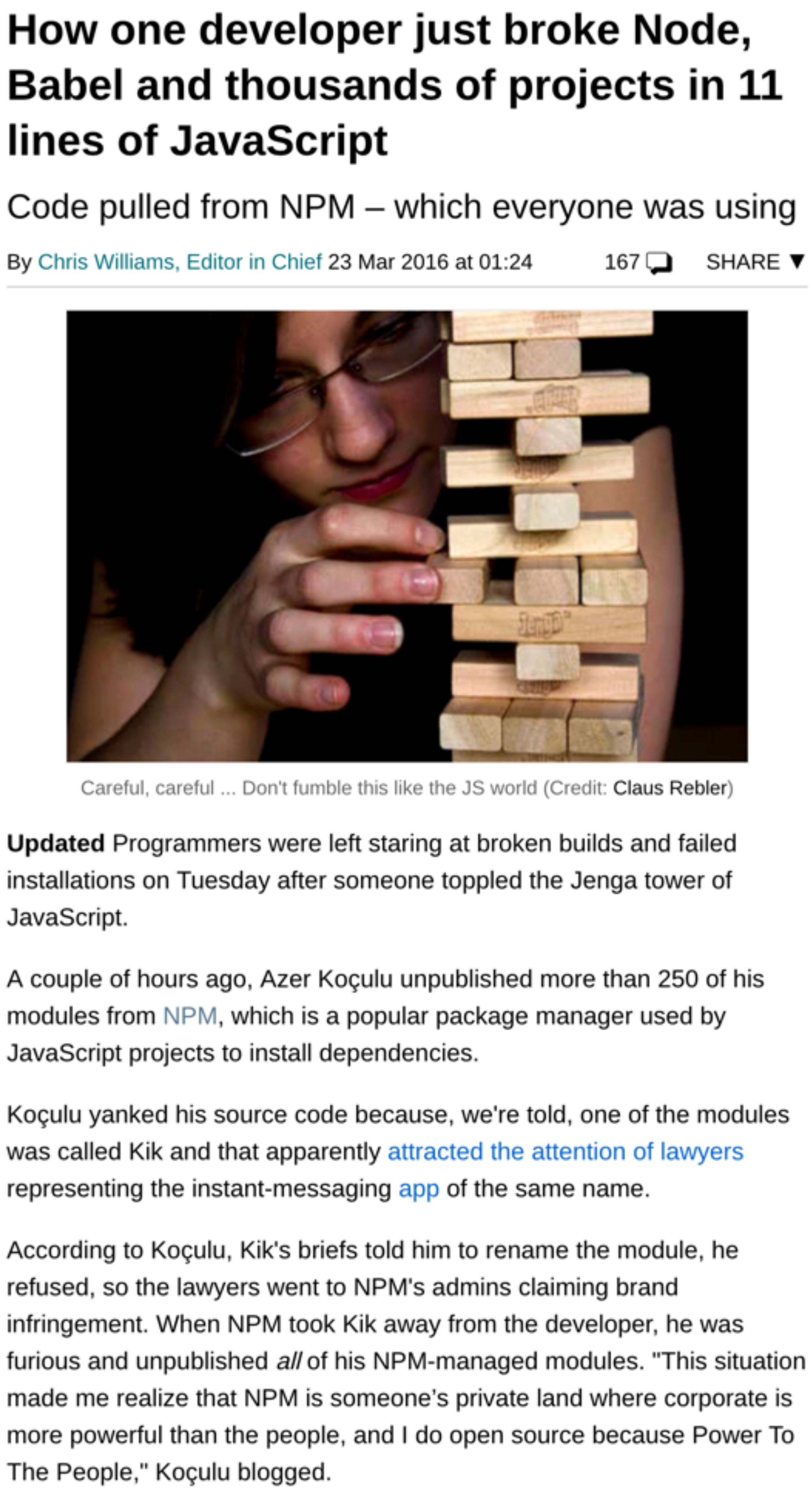
- Eclipse IDE has ecosystem of plugins that extend Eclipse to offer additional functionality (e.g., support for additional programming language)
- Eclipse Ecosystem values backwards compatibility
- Has tooling to identify unexpected subtle breaking changes
- Maintains backwards compatible interfaces
- Large effort put into release planning to ensure smooth transitions and infrequent releases

EXAMPLE: BREAKING CHANGES IN NPM ECOSYSTEM

- Demands little of developers making breaking change
- Ok to make any breaking change, just need to increment version correctly
  - Enables exploration of different API design to achieve better usability
- Downstream projects can specify which version of package to use
- No central release planning, individual package authors can make any changes they desire

EXAMPLE: LEFT-PAD IN NPM

- Developer removed 250 modules from NPM
- One of these was left-pad
  - Had 2,486,696 downloads in one month
- Downstream users now depended on module that no longer existed
- So disruptive that NPM violated community norms by bringing module back against wishes of author



How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript

Code pulled from NPM – which everyone was using

By Chris Williams, Editor in Chief 23 Mar 2016 at 01:24 167 SHARE ▼

Careful, careful ... Don't fumble this like the JS world (Credit: Claus Rebler)

Updated Programmers were left staring at broken builds and failed installations on Tuesday after someone toppled the Jenga tower of JavaScript.

A couple of hours ago, Azer Koçulu unpublished more than 250 of his modules from NPM, which is a popular package manager used by JavaScript projects to install dependencies.

Koçulu yanked his source code because, we're told, one of the modules was called Kik and that apparently attracted the attention of lawyers representing the instant-messaging app of the same name.

According to Koçulu, Kik's briefs told him to rename the module, he refused, so the lawyers went to NPM's admins claiming brand infringement. When NPM took Kik away from the developer, he was furious and unpublished *all* of his NPM-managed modules. "This situation made me realize that NPM is someone's private land where corporate is more powerful than the people, and I do open source because Power To The People," Koçulu blogged.

https://www.theregister.co.uk/2016/03/23/npm_left_pad_chaos/

RESPONDING TO UPSTREAM CHANGE

- When an upstream system changes, downstream system may have options about whether, when, how to respond
  - Ignore? Upgrade immediately? Upgrade later?

MONITORING CHANGE

- How do you know when an upstream system introduced change?
- Strategies
  - Actively monitor GitHub projects for systems (high effort)
  - Participate in upstream project: offers voice into features and functionality of project
  - Social awareness: follow twitter, mailing lists to learn about changes
  - Reactive monitoring: wait to hear about problems others experience
  - Testing: ensure that system works correctly

REDUCING EXPOSURE TO CHANGE

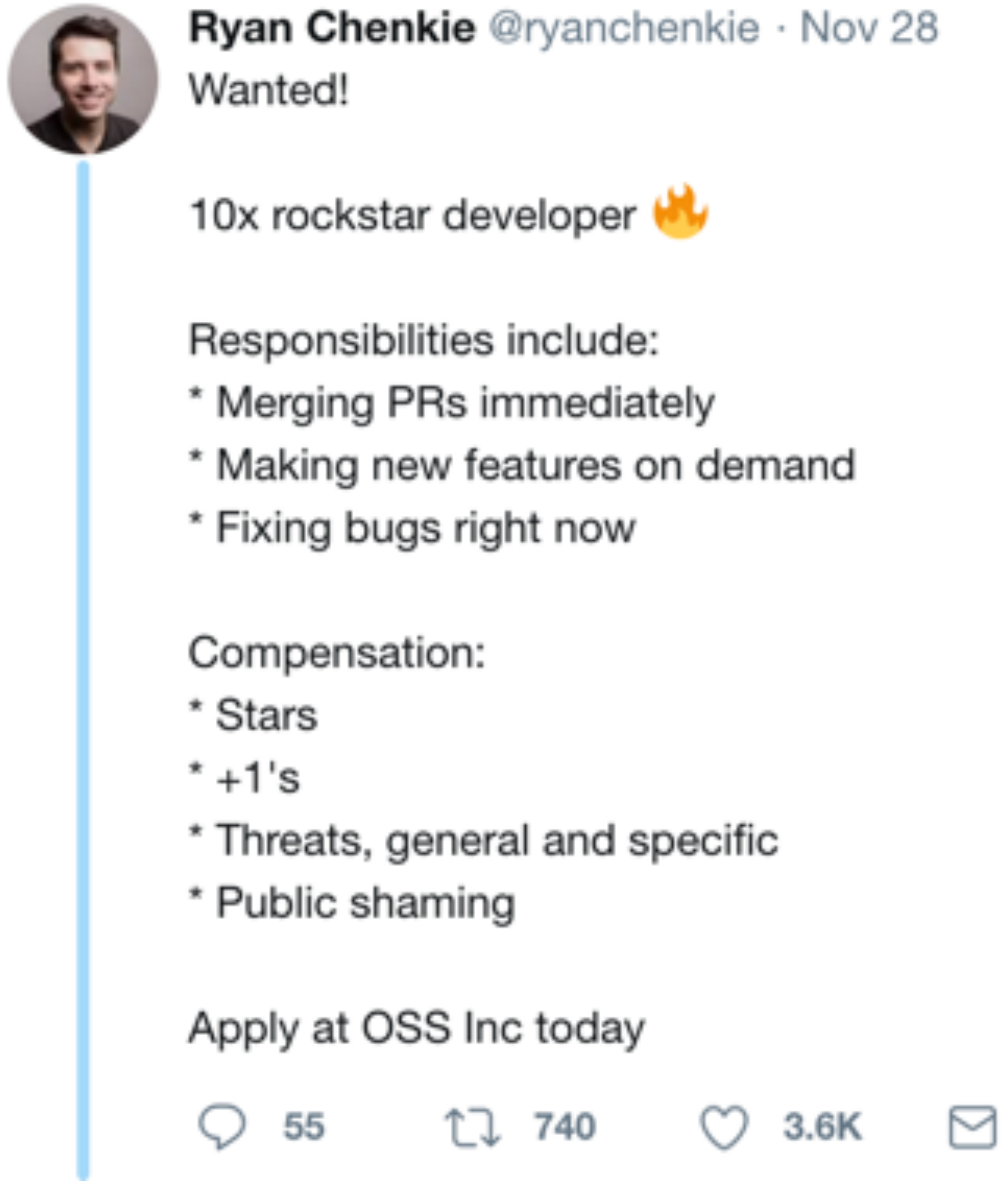
- Can reduce effort to monitor and react to change by reducing dependencies
- Only depend on things that are really important
- Copy or recreate functionality internally
- Selecting appropriate dependencies with signals that indicated high quality
    - Reputation from large organization or past success
    - Activity level of project
    - Size and identify of user base
    - Past history of dealing with changes
    - Quality of project artifacts: coding style, documentation, code size

POLICIES AND COMMUNITIES

- Within ecosystem, not everyone may act consistently with practices
    - Need to make a breaking change quickly, don't want to indicate through a major version change, which might be more work to adopt

MAINTAINER BURNOUT

- Growing problem in OSS community for packages which gain popularity which are **not** accusing value for key ecosystem player (e.g., Facebook, Twitter, Google)
- Someone built something which everyone uses and published it as OSS
- How are maintainer compensated?
- Who pays?



Ryan Chenkie @ryanchenkie · Nov 28

Wanted!

10x rockstar developer 🔥

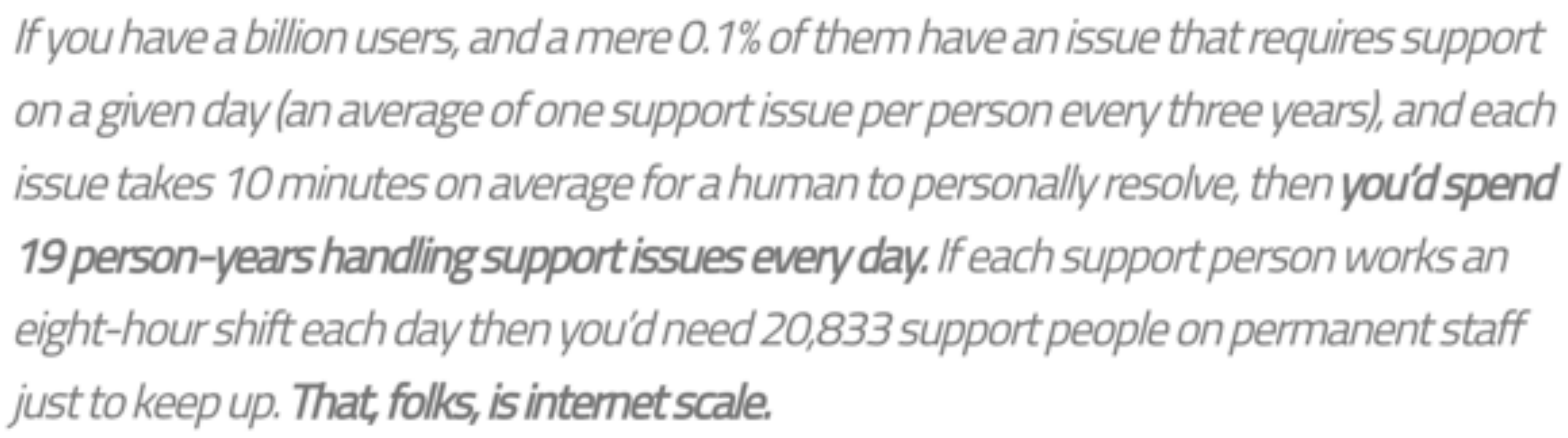
Responsibilities include:
* Merging PRs immediately
* Making new features on demand
* Fixing bugs right now

Compensation:
* Stars
* +1's
* Threats, general and specific
* Public shaming

Apply at OSS Inc today

https://twitter.com/ryanchenkie/status/1067801413974032385

MAINTAINER BURNOUT



If you have a billion users, and a mere 0.1% of them have an issue that requires support on a given day (an average of one support issue per person every three years), and each issue takes 10 minutes on average for a human to personally resolve, then ***you'd spend 19 person-years handling support issues every day.*** *If each support person works an eight-hour shift each day then you'd need 20,833 support people on permanent staff just to keep up.* ***That, folks, is internet scale.***

- In principle, can publish something once and have an infinite number of users at no additional cost
- Does not work in practice
- Maintainers may abandon project
- Open question: who should maintain abandoned projects?

SUMMARY

- Software systems exists in context of **ecosystem** of upstream and downstream systems connected by dependencies
- Ecosystems may be centered around OS, application, or end-user programming or distributed into individual packages
- Breaking changes incur costs, which can be distributed between upstream or downstream systems
- Different ecosystems have different values and policies for dealing with breaking changes

IN CLASS ACTIVITY

PART 1: CHARACTERIZE A SOFTWARE ECOSYSTEM

- In groups of 1, 2 or 3,
- Pick a software ecosystem which we did **not** discuss in class
  - Pick one that you or your group has used before
- Deliverables
  - Describe the software ecosystem: is it OS, application, or end-user programming centric; or is it distributed and package centric?
  - How does ecosystem handle breaking changes? How is this policy related to ecosystem's values?
- DUE: 12pm Friday, May 2

PART 2: INTRODUCE A BREAKING CHANGE

- Now imagine that you are a developer inside the organization at the center of your ecosystem
- You need to make a breaking change.
- How will you do this?
- Deliverables
  - Describe a (fictional) breaking change. What changed?
  - How will you mitigate the impact of this change?
- DUE: 12pm Friday, May 2