

# Perma- web

Neil Kandalgaonkar

2024-06-27 · Vancouver, Canada

Our Networks 2024

The background of the slide features a marbled pattern with swirling designs in shades of red, orange, yellow, and green on a light grey base. A solid blue rectangular area is centered on the slide, containing the text.

**A few years ago,  
I redid my personal website**

That binary number has a decimal equivalent of 2595

And that's how LetterPwn currently stands:

E	F	P	J	A
C	C	W	C	E
J	F	J	V	K
Q	M	M	P	B
K	U	M	L	T

Note that in this example, their binary is on the lower part of the board.

### Determining all the ways

So we've determined that *mumble* can have to find those squares again, and a few more. Once again, **bitmaps** to the rescue.

We map each letter to bitmaps of all possible **combinations**, we recursively produce the *mumble*. We have to make sure that we

E	F	P	J	A
C	C	W	C	E
J	F	J	V	K
Q	M	M	P	B
K	U	M	L	T

E	F	P	J	A
C	C	W	C	E
J	F	J	V	K
Q	M	M	P	B
K	U	M	L	T

This gives us a list of six moves we could play on the board, represented in bitmap form, so it's just a list of six integers:

11206657, 11207168, 15269889, 15278460, 15315425, 15335936

August 12th, 2012

## ToorCamp 2012

I attended [ToorCamp 2012](#), the "five day, open air, tech camping event" in Washington State.

We managed to find the one place in Washington state where it is freezing cold and damp in the summertime. I think it just touches the current that eventually makes the Bay Area so frigid.

**Things I would never do anywhere else:** program a robot to dance, hold a sword receiving blasts of purple lighting bolts from a Gauss gun, worship at the Church of Robotron, get a preview of a cool interactive toy [Star](#) was working on, help assemble a tower built of pipes to validate one of [Milos'](#) theories about constructing large structures with struts of uniform length, play foosball on a table that automatically keeps score and lights up when points are scored, attend a bicycle jousting event....



Incidentally, it is fun to terrorize small children with robots. Hacked on [MC Hawking's](#) software with some other NoiseBridge folks so it could be remotely controlled. Was a bit surprised that it was all in Java (it's some standard package for doing robots, apparently). That made consuming a line-oriented protocol super annoying, and my more unixy style of programming is a bit foreign to Java, but we eventually made it work.

**Conversations I would never have anywhere else:** transgender hacker still adapting to her new life, Egyptian activist on exactly how influential the internet was in the 2010-2011 revolution, how to set up your own GSM phone service

- Why isn't the phone network working yet?

- [complex explanation about crystals and synchronization]. This will take another six hours.

- Isn't there some quick and dirty way to do this?

- This is the quick and dirty way.

**Spontaneously taught:** Haskell, how unix process forking works

**Spontaneously learned:** lock picking, the chemistry of baking, how to turn kelp into a musical instrument, Bitcoin, cloning, LASERS

It's the kind of conference where you overhear a woman talking about her sex toy experiments and how nobody else seems to do this, and not one but two bystanders inform her she should be talking to [qDot](#).



39

I helped make that thing you like on the Internet. Google, Upcoming, Flickr, and Wikipedia. More info...

### Recent Posts

[Answer to Ed Felten's Hello World puzzle](#)

[Manliness for manly men like me](#)

[HeartForth](#)

[Thoughts about OuterNet](#)

[Less represented and loving it](#)

Tweets by @flipzagger

maker of things

39

I helped make that thing you like on the Internet. Google, Upcoming, Flickr, and Wikipedia. More info...

### Recent Posts

[Answer to Ed Felten's Hello World puzzle](#)

[Manliness for manly men like me](#)

[HeartForth](#)

[Thoughts about OuterNet](#)

[Less represented and loving it](#)

Tweets by @flipzagger



great people jumped in to help. I started working with [\(mnoisse\)](#) to make this happen. Casey was attracted to this, and I focused on all the R&D and planning. We got support from the [Ardent](#) community, and built the whole

this seems like a simple project, but there were a hundred little details. How to communicate to Burners what you were even doing with it? Exactly how should it be oriented to maximize the visibility? Ensuring that people can't climb on it, while also making it easy to move. What medium should we use to hold the beads? And how do

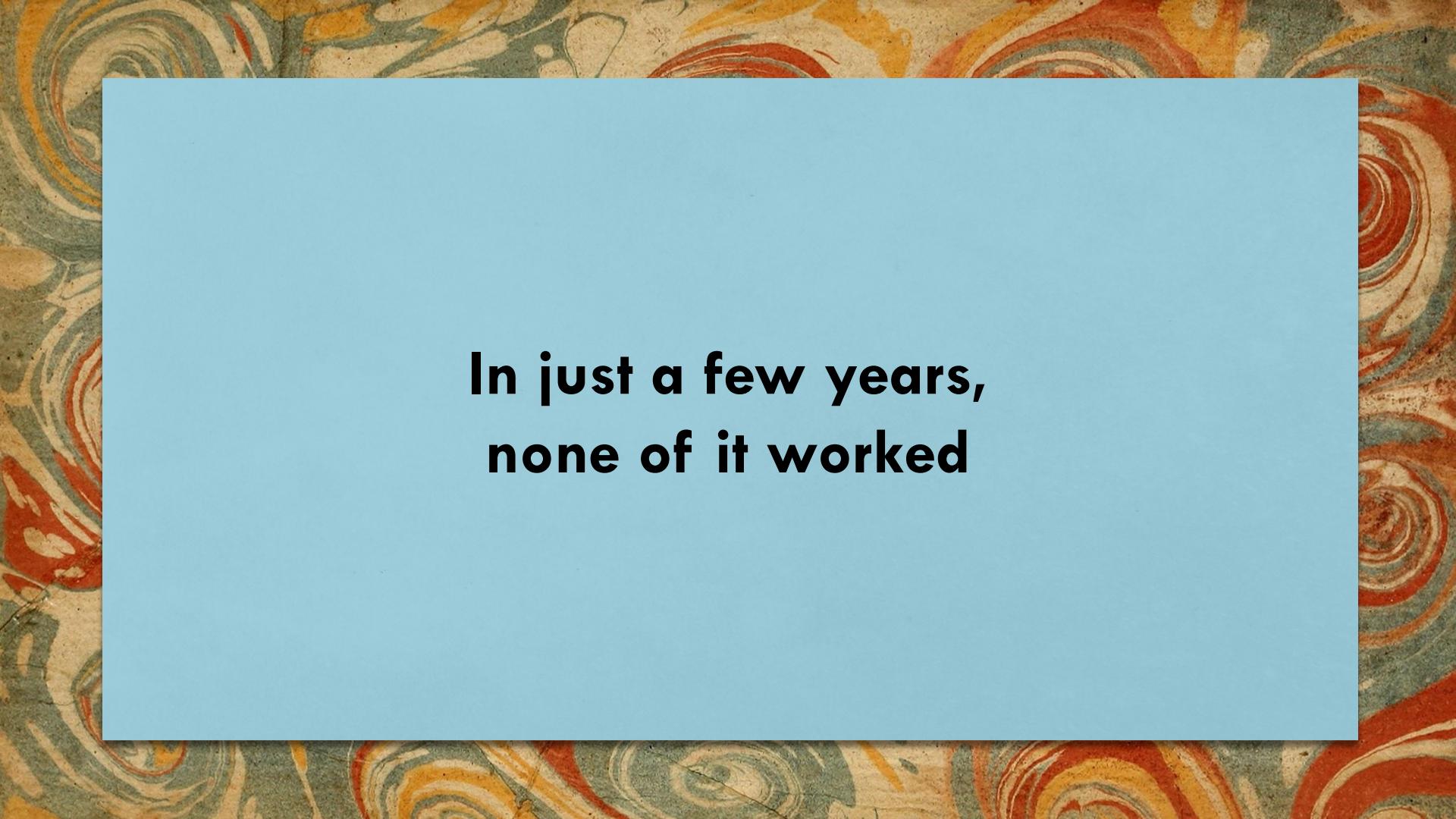
we protect that surface?

Some of my favorite moments were during build season. When I showed the first working prototype to [Audrey](#), she exclaimed that it was going to freak

# Hacker-friendly choices

## Octopress - static generator

- Jekyll
  - Markdown/YAML
- Ruby
- CSS, SCSS, LESS
- Octopress extensions
  - Pygments; syntax coloring
  - Octopress-Flickr; hosted images
  - Fancybox.js
- Official Twitter Javascript API

The background of the image features a marbled pattern with swirling designs in shades of red, orange, yellow, and green on a light-colored surface.

**In just a few years,  
none of it worked**

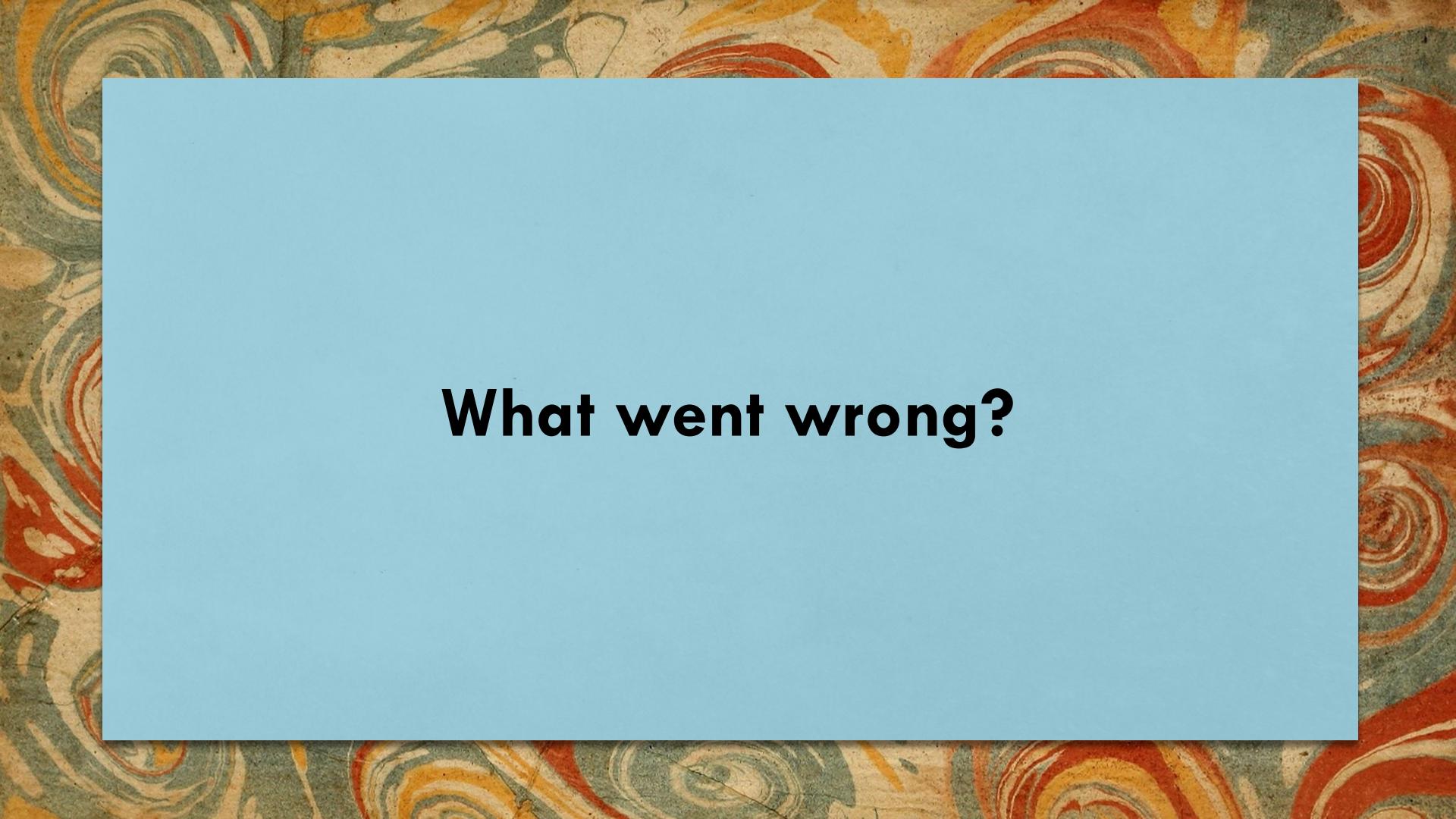
# Hacker-friendly choices?

## Octopress - static generator

- Jekyll
  - Markdown/YAML
- Ruby
- CSS, SCSS, LESS
- Octopress extensions
  - Pygments; syntax coloring
  - Octopress-Flickr; hosted images
- Fancybox.js
- Official Twitter Javascript API

# Hacker-friendly choices?





**What went wrong?**

Everything will break  
...and sooner than you expect

**“Permacomputing”** (from [permacomputing.net](http://permacomputing.net))

Have few dependencies

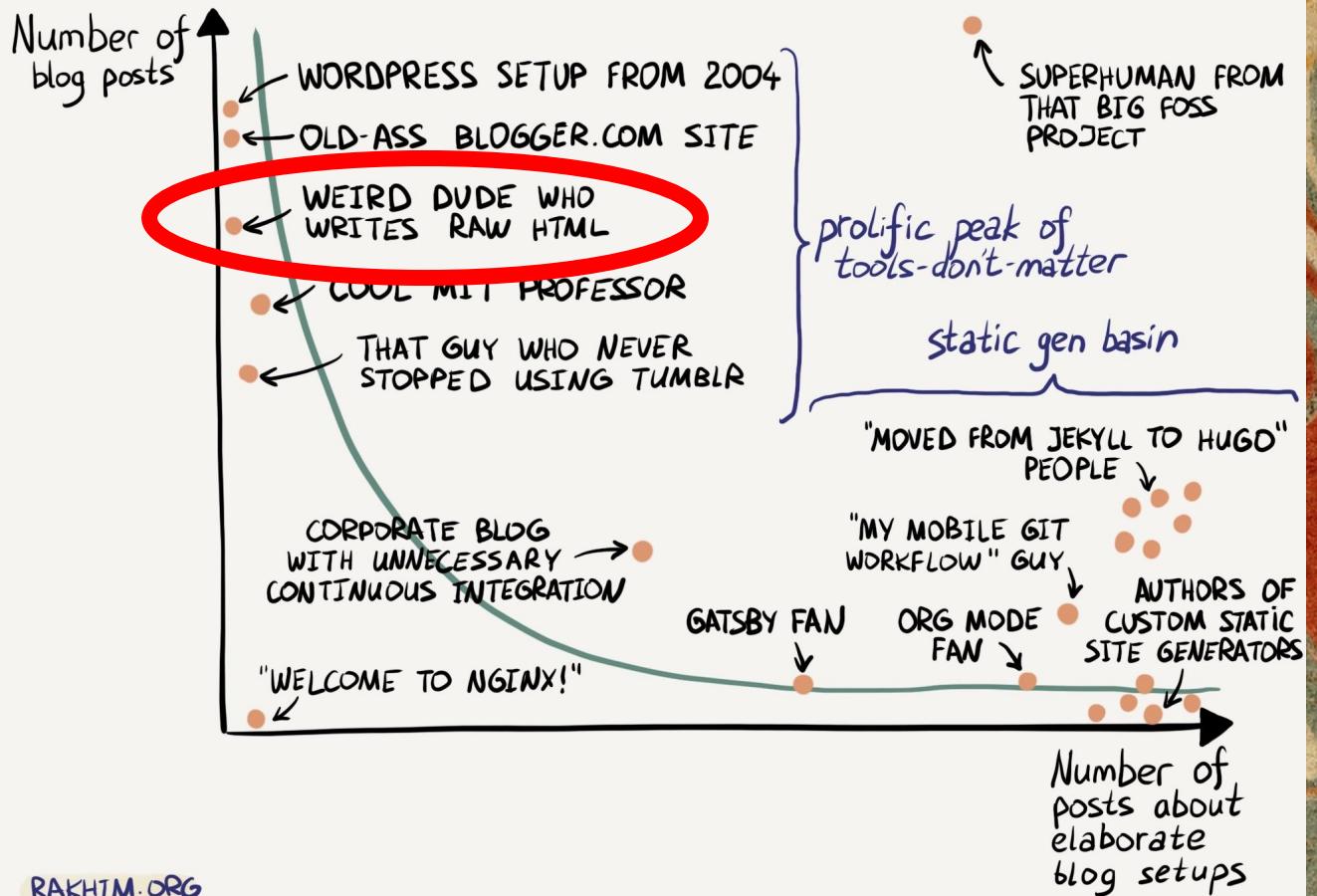
Avoid pseudosimplicity

**Accumulate wisdom and experience  
rather than codebase**

Human-scale

Hope for the best; prepare for the worst

**Design for descent (degrade gracefully)**



The container throttling problem | Patreon

This is an excerpt from an internal document David Mackey and I co-authored in April 2019. The document is excerpted since much of the original doc was about comparing possible approaches to increasing efficiency at Twitter, which is mostly information that's meaningless outside of Twitter without a large amount of additional explanation/context.

At Twitter, most CPU bound services start failing over at around 50% reserved container CPU utilization and almost all services start failing over at not much more CPU utilization even though CPU bound services should, theoretically, be able to get higher CPU utilizations. Because load isn't, in general, evenly balanced across shards and the shard-level degradation in performance is so severe when we exceed 50% CPU utilization, this makes the practical limit much lower than 50% even during peak load events.

This document will describe potential solutions to this problem. We'll start with describing why we should expect this problem given how services are configured and how the Linux scheduler we're using works. We'll then look into case studies of how we can fix this with config tuning for specific services, which can result in a 1.5x to 2x increase in capacity, which can translate into \$[redacted]M/yr to \$[redacted]M/yr in savings for large services. While this is worth doing and we might get back \$[redacted]M/yr to \$[redacted]M/yr in TCO by doing this for large services, manually fixing services one at a time isn't really scalable, so we'll also look at how we can make changes that can recapture some of the value for most services.

### The problem, in theory

Almost all services at Twitter run on Linux with the CFS scheduler, using CFS bandwidth control quota for isolation, with default parameters. The intention is to allow different services to be colocated on the same boxes without having one service's runaway CPU usage impact other services and to prevent services on empty boxes from taking all of the CPU on the box, resulting in unpredictable performance, which service owners found difficult to reason about before we enabled quotas. The quota mechanism limits the amortized CPU usage of each container, but it doesn't limit how many cores the job can use at any given moment. Instead, if a job "wants to" use more than that many cores over a quota timeslice, it will use more cores than its quota for a short period of time and then get throttled, i.e., basically get put to sleep, in order to keep its amortized core usage below the quota, which is disastrous for tail latency.

Since the vast majority of services at Twitter use thread pools that are much larger than their mesos core reservation, when jobs have heavy load, they end up requesting and then using more cores than their reservation and then throttling. This causes services that are provisioned based on load test numbers or observed latency under load to over provision CPU to avoid violating their SLOs. They either have to ask for more CPUs per shard than they actually need or they have to increase the number of shards they use.

An old example of this problem was the JVM Garbage Collector. Prior to work on the JVM to make the JVM container aware, each JVM would default the GC parallel thread pool size to the number of cores on the machine. During a GC, all these GC threads would run simultaneously, exhausting the cpu quota rapidly causing throttling. The resulting effect would be that a subsecond stop-the-world GC pause could take many seconds of wallclock time to complete. While the GC issue has been fixed, the issue still exists at the application level for virtually all services that run on mesos.

### The problem, in practice [case study]

As a case study, let's look at `service-1`, the largest and most expensive service at Twitter.

Below is the CPU utilization histogram for this service just as it starts failing its load test, i.e., when it's just above the peak load the service can handle before it violates its SLO. The x-axis is the number of CPUs used at a given point in time and the y-axis is (relative) time spent at that utilization. The service is provisioned for 20 cores and we can see that the utilization is mostly significantly under that, even when running at nearly peak possible load:

## CPU Utilization Histogram

displays a percentage of the wall time the specific number of CPUs were running



The figure shows a histogram of CPU utilization. The x-axis is labeled 'CPUs' and ranges from 0 to 20. The y-axis is labeled 'Relative Log Time' and ranges from 0 to 1.0. The histogram bars are blue. The distribution is heavily skewed towards lower utilization levels, with the highest frequency occurring at 1 CPU. As the number of CPUs increases, the relative log time decreases, indicating that higher utilization levels are less frequent over time. A vertical dashed line is drawn at 20 CPUs, labeled 'Average Load'. The histogram shows that while most of the time is spent at low utilization levels, there are significant spikes reaching up to 20 CPUs, which is the average load.

# Can a “permacomputing” site look nice?

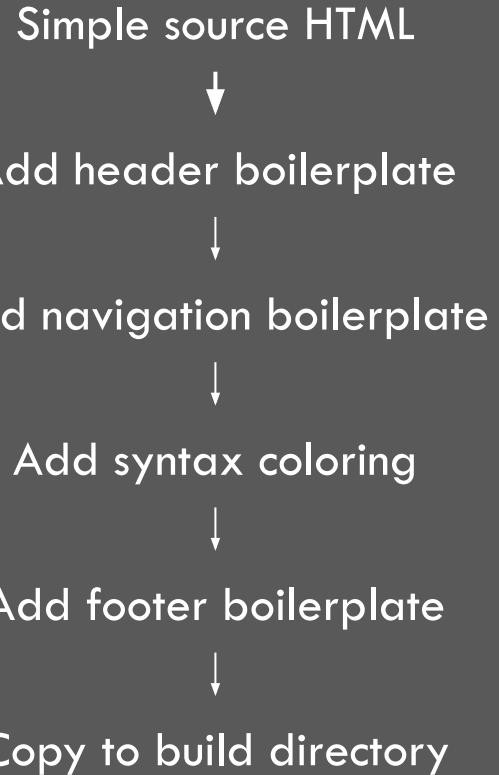
Write simple, raw HTML

...automatically transform it to better HTML

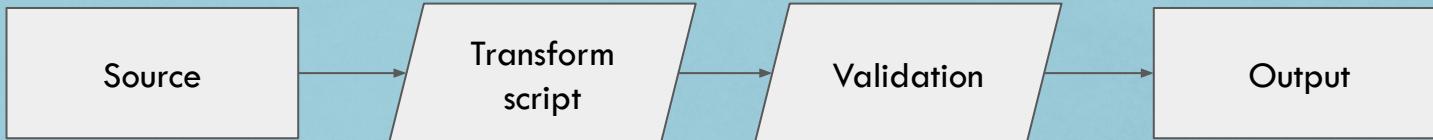
...in stages

...if any transformation stage breaks, the original HTML is passed to the next phase

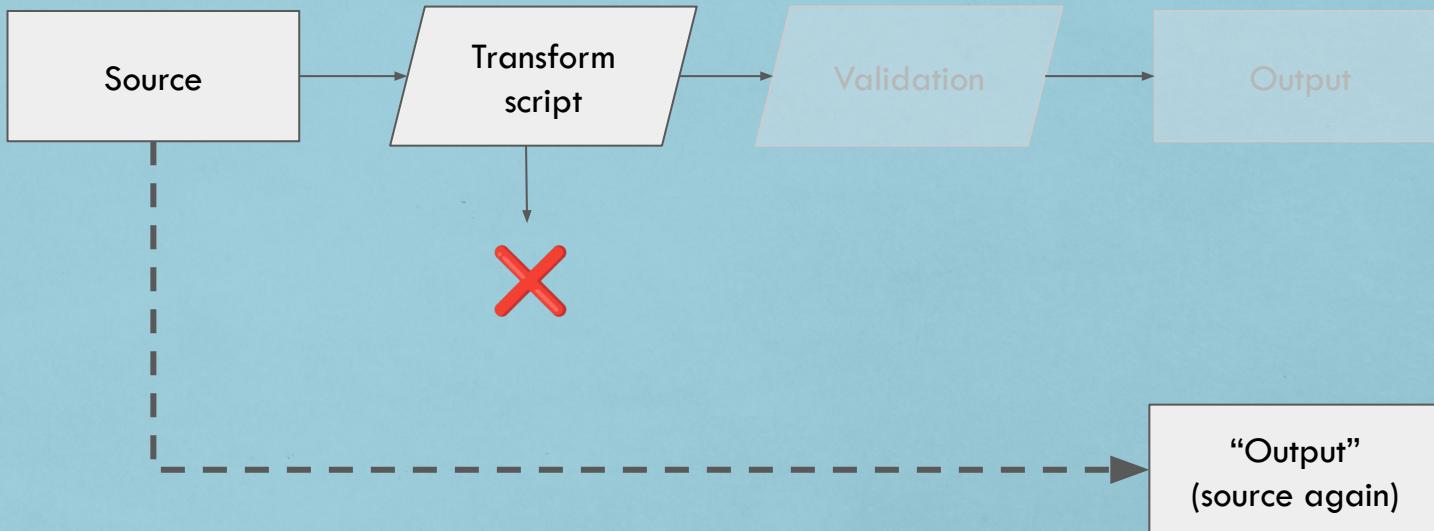
# Failure embracing HTML transformation pipeline



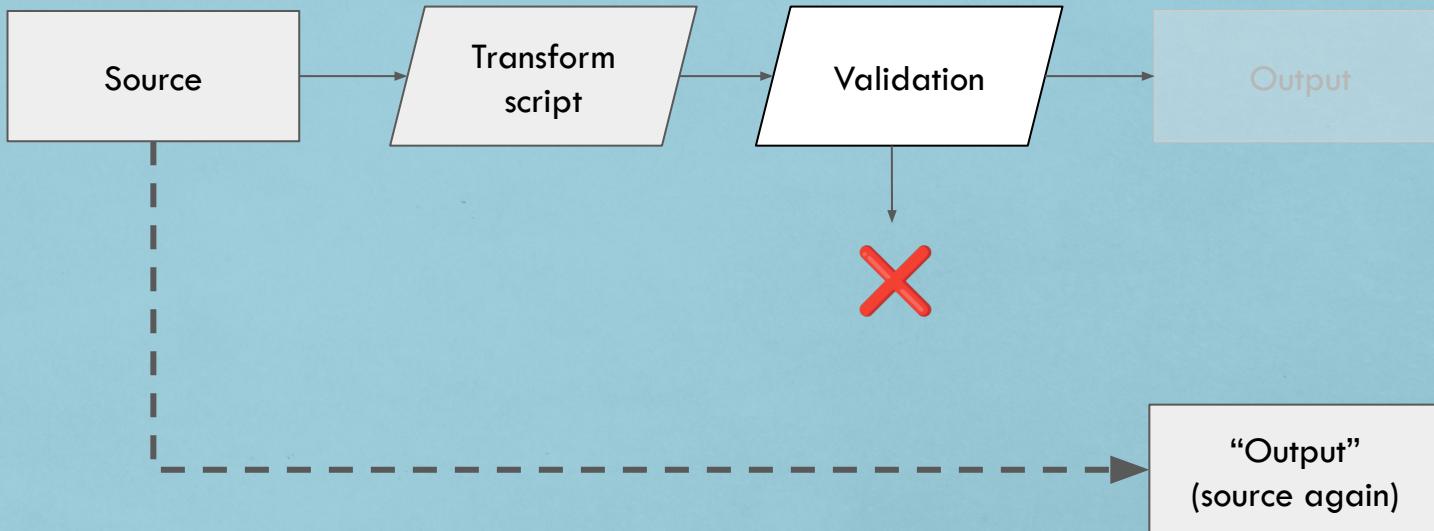
# How to make each step fail forward



# How to make each step fail forward



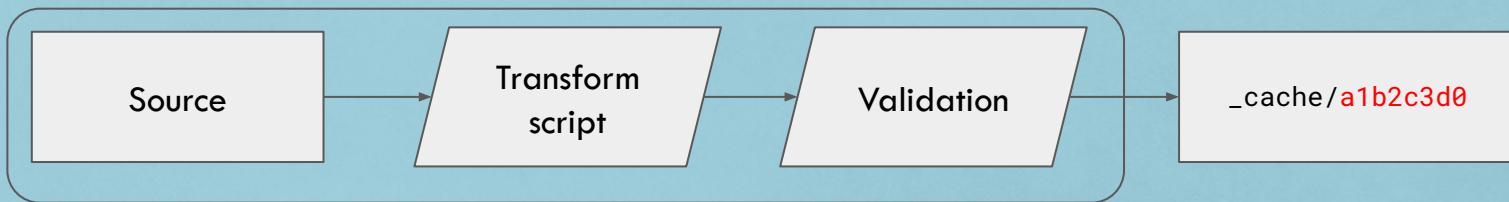
# How to make each step failure-free



Fun fact: we're doing this trick with hardlinks

# How to make each step very fast

Use a content-addressable cache



Hashes to **a1b2c3d0**

If source, script, and validation are the same we skip it, and reuse the cached content. Most changes result in very little reprocessing.

## **What technologies to use?**

For any technology, assume you're somewhere in the middle of its lifespan

React.js has been around for 10 years, so you can assume it will be around for 10 more

What if you want it to last for 20 years? 30 years? Your whole lifetime?

# 30 year tech

The core engine is just *make*  
and *bash* running on a Unix.  
Ancient computing concepts.

Very small engine.



**BUT:** each transformation script can  
be the latest and greatest.  
JavaScript, Python...

*If it breaks, just replace it*

*If you don't get around to replacing  
it, no problem*

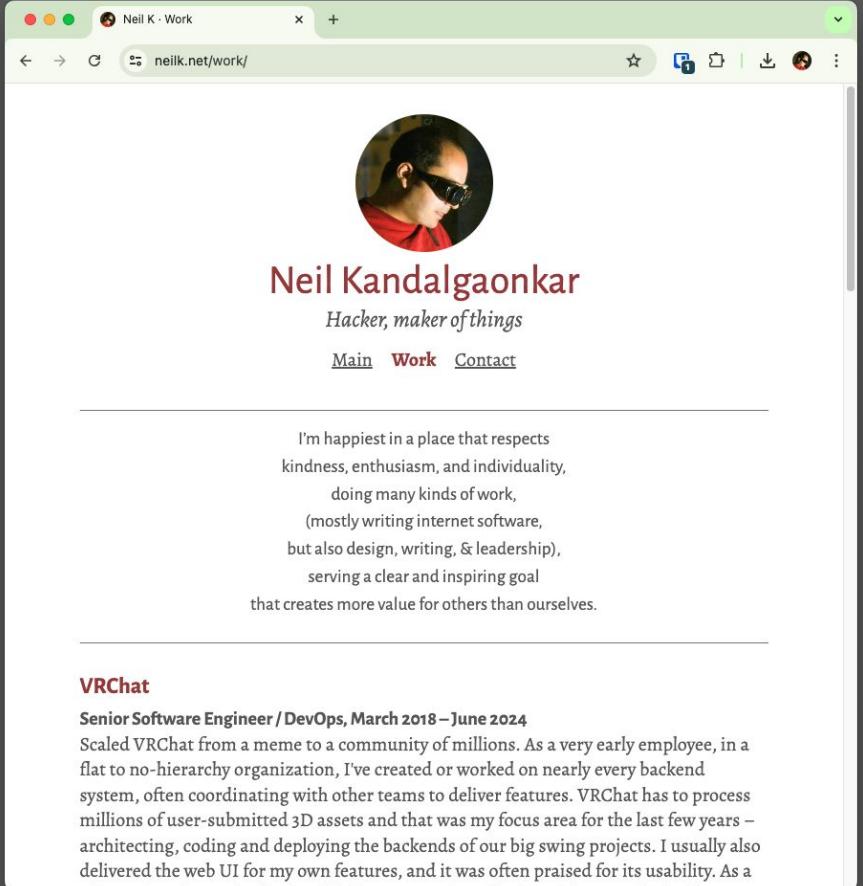
*The site still works and you can still  
publish*



# My personal website's CV looks like this

(Hint: I'm looking for opportunities...)

# Today



The screenshot shows a web browser window titled "Neil K - Work" with the URL "neilk.net/work/". The page features a circular profile picture of a man wearing sunglasses and a red shirt. Below the picture, the name "Neil Kandalgaonkar" is displayed in a large, bold, dark red font, followed by the subtitle "Hacker, maker of things" in a smaller, italicized dark red font. A navigation bar below includes links for "Main", "Work", and "Contact". A horizontal line separates this from the main content area. In the main content area, there is a block of text describing the author's philosophy: "I'm happiest in a place that respects kindness, enthusiasm, and individuality, doing many kinds of work, (mostly writing internet software, but also design, writing, & leadership), serving a clear and inspiring goal that creates more value for others than ourselves." Another horizontal line separates this from the next section. The section title "VRChat" is in bold black font, followed by the subtitle "Senior Software Engineer / DevOps, March 2018 – June 2024" in a smaller black font. The text below describes the author's role at VRChat, mentioning their involvement in scaling the platform from a meme to a community of millions, working on various backend systems, coordinating with other teams, and architecting, coding, and deploying features. It also notes their focus on user-submitted 3D assets and the web UI for their own features, which was praised for its usability.

Neil Kandalgaonkar

Hacker, maker of things

Main Work Contact

---

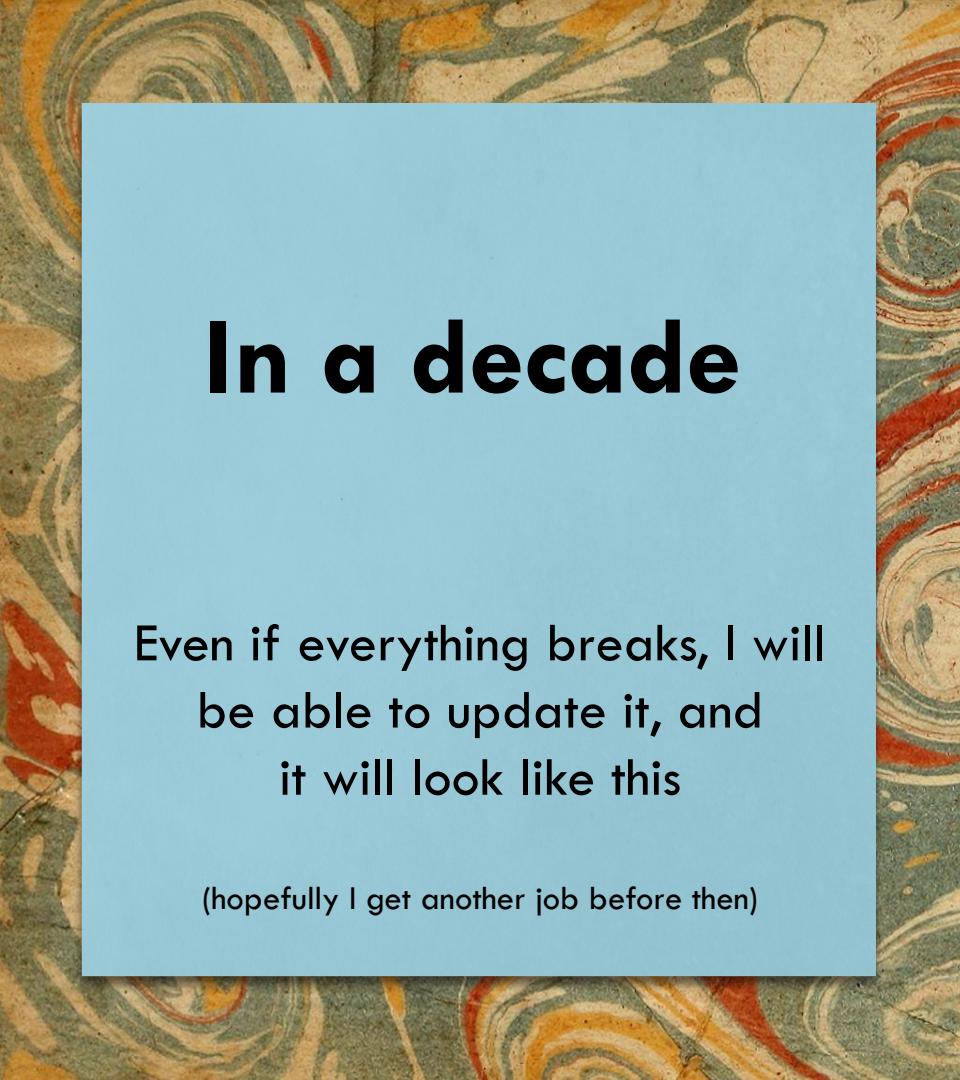
I'm happiest in a place that respects  
kindness, enthusiasm, and individuality,  
doing many kinds of work,  
(mostly writing internet software,  
but also design, writing, & leadership),  
serving a clear and inspiring goal  
that creates more value for others than ourselves.

---

**VRChat**

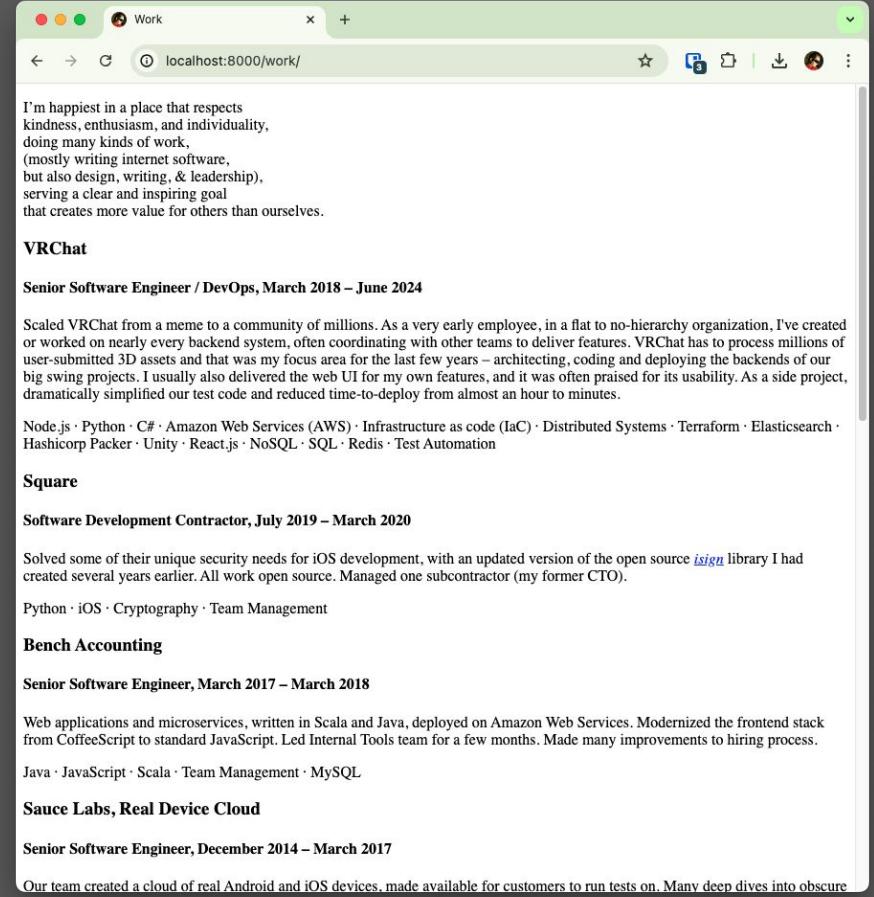
Senior Software Engineer / DevOps, March 2018 – June 2024

Scaled VRChat from a meme to a community of millions. As a very early employee, in a flat to no-hierarchy organization, I've created or worked on nearly every backend system, often coordinating with other teams to deliver features. VRChat has to process millions of user-submitted 3D assets and that was my focus area for the last few years – architecting, coding and deploying the backends of our big swing projects. I usually also delivered the web UI for my own features, and it was often praised for its usability. As a



In a decade  
Even if everything breaks, I will  
be able to update it, and  
it will look like this

(hopefully I get another job before then)



I'm happiest in a place that respects kindness, enthusiasm, and individuality, doing many kinds of work, (mostly writing internet software, but also design, writing, & leadership), serving a clear and inspiring goal that creates more value for others than ourselves.

**VRChat**

**Senior Software Engineer / DevOps, March 2018 – June 2024**

Scaled VRChat from a meme to a community of millions. As a very early employee, in a flat to no-hierarchy organization, I've created or worked on nearly every backend system, often coordinating with other teams to deliver features. VRChat has to process millions of user-submitted 3D assets and that was my focus area for the last few years – architecting, coding and deploying the backends of our big swing projects. I usually also delivered the web UI for my own features, and it was often praised for its usability. As a side project, dramatically simplified our test code and reduced time-to-deploy from almost an hour to minutes.

Node.js · Python · C# · Amazon Web Services (AWS) · Infrastructure as code (IaC) · Distributed Systems · Terraform · Elasticsearch · Hashicorp Packer · Unity · React.js · NoSQL · SQL · Redis · Test Automation

**Square**

**Software Development Contractor, July 2019 – March 2020**

Solved some of their unique security needs for iOS development, with an updated version of the open source *isign* library I had created several years earlier. All work open source. Managed one subcontractor (my former CTO).

Python · iOS · Cryptography · Team Management

**Bench Accounting**

**Senior Software Engineer, March 2017 – March 2018**

Web applications and microservices, written in Scala and Java, deployed on Amazon Web Services. Modernized the frontend stack from CoffeeScript to standard JavaScript. Led Internal Tools team for a few months. Made many improvements to hiring process.

Java · JavaScript · Scala · Team Management · MySQL

**Sauce Labs, Real Device Cloud**

**Senior Software Engineer, December 2014 – March 2017**

Our team created a cloud of real Android and iOS devices, made available for customers to run tests on. Many deep dives into obscure

The end

Neil Kandalgaonkar  
<https://neilk.net/>

Source code (eventually)  
<https://github.com/neilk/permaweb>

# Image credits

## Wikimedia Commons

- [https://commons.wikimedia.org/wiki/File:Histoire\\_g%C3%A9n%C3%A9rale\\_des\\_auteurs\\_sacr%C3%A9s\\_et\\_eccl%C3%A9siastiques\\_-\\_qui\\_contient\\_leur\\_vie,\\_le\\_catalogue,\\_la\\_critique,\\_le\\_jugement,\\_la\\_chronologie,\\_l%27analyse\\_and\\_le\\_d%C3%A9nombrement\\_des\\_diff%C3%A9rentes\\_%C3%A9ditions\\_de\\_leurs\\_\(14597796317\).jpg](https://commons.wikimedia.org/wiki/File:Histoire_g%C3%A9n%C3%A9rale_des_auteurs_sacr%C3%A9s_et_eccl%C3%A9siastiques_-_qui_contient_leur_vie,_le_catalogue,_la_critique,_le_jugement,_la_chronologie,_l%27analyse_and_le_d%C3%A9nombrement_des_diff%C3%A9rentes_%C3%A9ditions_de_leurs_(14597796317).jpg)
- [https://commons.wikimedia.org/wiki/File:Histoire\\_g%C3%A9n%C3%A9rale\\_des\\_auteurs\\_sacr%C3%A9s\\_et\\_eccl%C3%A9siastiques\\_-\\_qui\\_contient\\_leur\\_vie,\\_le\\_catalogue,\\_la\\_critique,\\_le\\_jugement,\\_la\\_chronologie,\\_l%27analyse\\_and\\_le\\_d%C3%A9nombrement\\_des\\_diff%C3%A9rentes\\_%C3%A9ditions\\_de\\_leurs\\_\(14800512523\).jpg](https://commons.wikimedia.org/wiki/File:Histoire_g%C3%A9n%C3%A9rale_des_auteurs_sacr%C3%A9s_et_eccl%C3%A9siastiques_-_qui_contient_leur_vie,_le_catalogue,_la_critique,_le_jugement,_la_chronologie,_l%27analyse_and_le_d%C3%A9nombrement_des_diff%C3%A9rentes_%C3%A9ditions_de_leurs_(14800512523).jpg)
- [https://commons.wikimedia.org/wiki/File:Old\\_paper7.jpg](https://commons.wikimedia.org/wiki/File:Old_paper7.jpg)
- [https://commons.wikimedia.org/wiki/File:Francisco\\_de\\_Goya,\\_El\\_sue%C3%B1o\\_de\\_la\\_razon\\_produce\\_mons\\_truos\\_\(The\\_Sleep\\_of\\_Reason\\_Produces\\_Monsters\),\\_published\\_1799,\\_NGA\\_7502.jpg](https://commons.wikimedia.org/wiki/File:Francisco_de_Goya,_El_sue%C3%B1o_de_la_razon_produce_mons_truos_(The_Sleep_of_Reason_Produces_Monsters),_published_1799,_NGA_7502.jpg)