# Sifting Technologies - Separating the Wheat From the Chaff

Nathaniel T. Schutta
@ntschutta
ntschutta.io

Technology changes.

Constantly.

Feature not a bug.

Keeps things interesting…

We want to avoid legacy platforms.

But we can’t change
things every few months.

“Our app has 4 different UI frameworks…”

Developers kept chasing the new hotness.

# How do we avoid that?

# How do we evaluate new technology?

I have no idea what language/ framework/platform is “next”.

No one does.

But I can guarantee you this much:

It will be different than
what we use today.

Five years from now we will be using something that isn’t invented yet.

# What do we do about that?

Evaluating technology is a key skill.

Chasing the new thing.

Technology changes.

Constantly.

Tempting to always chase the “new thing”.

Bleeding edge.

It’s fun!

Part of being in this industry.

Our understanding constantly evolves.

Twitter, Inc.

# Thread

**Benedict Evans**
@BenedictEvans

There are things you understand well enough to have strong opinions on, things you know just well enough to know what others' opinions are, and things where you can have no opinion. The complication in tech is that things move between these categories all the time.

1:42 PM · Dec 30, 2017

**70** Retweets **218** Likes

**Benedict Evans** @BenedictEvans · Dec 30
Replying to @BenedictEvans
In tech, there are:
Things you're starting to learn about
Things you know well
Things you're no longer following
Things you were an expert on but moved on from years ago and half-forgot

In tech, the point that you really understand something is the point it's becoming irrelevant

5 79 193

**Benedict Evans** @BenedictEvans · Dec 30
For example, I used to know all about radio networks, mobile networks and the Japanese mobile market iMode! IAppli! PHS!). Now, none of that matters. Most questions about smartphones are now moving rapidly down the list as well. On to new questions.

13 2 50

@FemiAbodunde · Dec 30
Replying to @BenedictEvans
🔥🔥🔥🔥The speed of the feed.

Let’s be honest…

Developers have opinions!

Often ⁺very⁺ strong opinions.

Maybe we fear old things?

Twitter, Inc.

# Thread

**Jeff Johnson (Mac/iOS developer for hire)** @lapcats... · Dec 21

It's said that "senior" developers tend to be afraid of learning new things. This has always struck me as absurd, because who hasn't learned a ton of new things over the course of 10+ years of programming? Almost impossible not to.

4 5 18

**Jeff Johnson (Mac/iOS developer for hire)** @lapcats... · Dec 21

It almost feels as though less experienced developers are projecting their own fears onto others with more experience.

1 1 8

**Jeff Johnson (Mac/iOS developer for hire)** @lapcats... · Dec 21

I get the sense that many programmers are afraid to learn *old* things. As if the older things are impossibly arcane and complex, compared to the supposed simplicity and elegance of new tech.

1 6 18

**Jeff Johnson (Mac/iOS developer for hire)** @lapcats... · Dec 21

They have a strong preference for the new. They'd rather use unfinished, buggy, unproven technology than older, refined, stable, tested tech. Even if it reinvents or indeed breaks the wheel.

2 7 15

**Jeff Johnson (Mac/iOS developer for hire)** @lapcats... · Dec 21

So now there's a of excitement about a "UnicornKit" that combined UIKit and AppKit into one. But do people realize that unless UnicornKit is simply UIKit, you're going to have to learn something else anyway?

1 3 9

**Jeff Johnson (Mac/iOS developer for hire)** @lapcats... · Dec 21

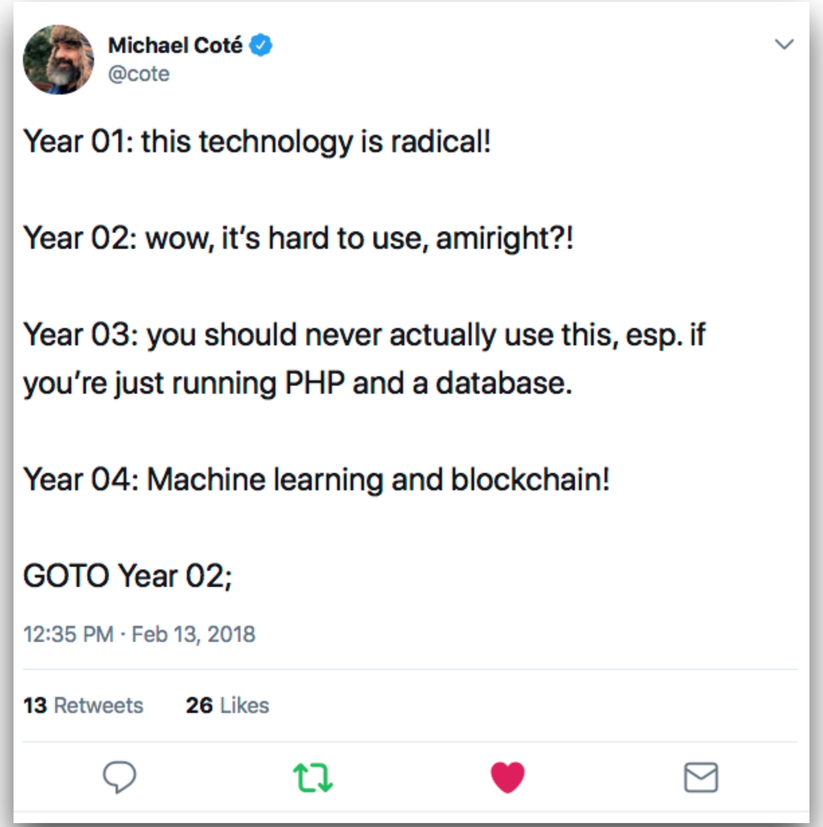
Why is learning UnicornKit harder than learning AppKit? Many developers have successfully written both Mac and iOS apps. It's not as hard as you may think if you have no experience with AppKit.

2 3 8

**Jeff Johnson (Mac/iOS developer for hire)** @lapcats... · Dec 21

Also, any new technology is going to be chock full of bugs. AppKit has its bugs, yes, but it's assuredly vastly more reliable and stable than UnicornKit

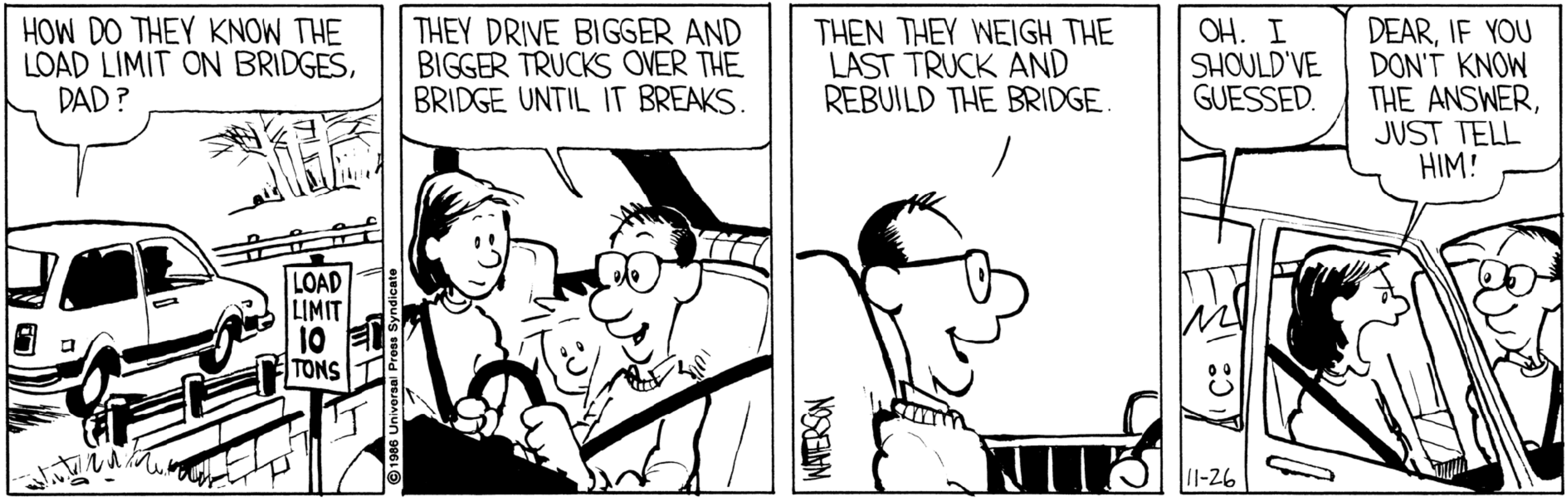
Predictable hype cycle.




https://mobile.twitter.com/cote/status/963481741171265537

How do we know where **not** to use a technology?

Trial and error.




CALVIN AND HOBBES © 1986 Watterson. Reprinted with permission of ANDREWS MCMEEL SYNDICATION. All rights reserved.

Developers tend to get bored quickly.

Learning keeps it fresh.

But we have to deliver business value.

Can’t do that if we’re always experimenting.

Have to commit at some point.

Develop some expertise.

Bleeding edge…means you will bleed!

Pioneers…the ones with arrows in their backs.

# What is your strategy?

# How do we avoid dead platforms?

Without constantly changing direction?

# Strategy.

Hope is not a strategy!

But it is what rebellions are built on.

We need to be deliberate.

There are a lot of bits out there…

New languages, techniques, approaches.

# How do you keep up?

Blogs? Books? Twitter?
Podcasts? Conferences?

Develop a routine.

Block out Friday afternoon.
Tuesday over lunch. Whatever fits.

Consider “morning coffee”.

Take 15-30 minutes in the morning to peruse the tech news.

Before the day gets away from you…

Attention is precious.

“Attention is a bit like real estate, in that they're not making any more of it. Unlike real estate, though, it keeps going up in value.”

— Seth Godin

http://sethgodin.typepad.com/seths_blog/2011/07/paying-attention-to-the-attention-economy.html

Don’t waste it.

Be selective.

Can’t read it all.

In fact, you'll miss almost everything.

http://www.npr.org/blogs/monkeysee/2011/04/21/135508305/the-sad-beautiful-fact-that-were-all-going-to-miss-almost-everything

We cannot adopt every new thing.

How do we know where to invest our time?

# Hacker's Radar?

http://www.paulgraham.com/javacover.html

“I have a hunch that [Java] won't be a very successful language.”

Never written a line of Java,
glanced at some books.

Need more than just a hunch.

“Judging Covers” can be a useful filter.

But beware bias.

# Where is the community?

Are you skating to
where the puck *was* ?

# Technology Radar.

https://www.thoughtworks.com/radar

thoughtworks.com

ESPAÑOL PORTUGUÊS DEUTSCH 中文 Search thoughtworks.com NEWS CONTACT US

ThoughtWorks

Clients Services Products Insights Events About us Careers

# TECHNOLOGY RADAR *VOL.17*

Insights into the technology and trends shaping the future

***Select an area to explore***

Search About the Radar Build your Radar

Downloads (EN, CA, ES, IT, PT, TR, 中文)

Subscribe to Technology Radar

## THEMES FOR THIS EDITION

### Open Source on the Rise in China

Because of changes in both attitude and policy, many large Chinese companies such as Alibaba and Baidu are rapidly releasing open source

### Kubernetes, the Choice for Container Orchestrator

The software development ecosystem is settling on Kubernetes and related

### Cloud as the New Normal

As cloud providers become more capable and reach parity of features, the public cloud model is becoming the new default for many organizations

### Trust in Blockchain More Evenly Distributed

Many of our clients are finding ways to leverage blockchain solutions for distributed ledgers and

Remember Google's 20% time?

Fallen out of favor in some circles…

# Innovation Fridays.

Could you carve out Friday afternoons?

# How about Tuesday Tech Talks?

# Architectural Briefings.

https://github.com/stuarthalloway/presentations/wiki/Architectural-Briefings

One person does some research, presents to the team.

And no, you don’t need to be an architect to present!

# Why should we use X?

What do you need to know to answer the “why”?

What do you need to know in order to use X?

Keep it short - 45 minutes.

Not a how to.

# Beyond the initial documentation.

Consider a basic template for the research…

# What is X?

# What problems does it solve?

# Is it simple?

# Where does it make sense to use it?

# Where *doesn't* it make sense to use it?

# What are the tradeoffs?

When we find ourselves presented with technology that promises to offer us drastic improvements, we need to look for the trade-offs.

— Susan J. Fowler
Production-Ready Microservices

# What are the alternatives?

# What is the cost?

Cost is multi-faceted.

What does it cost to get started? Training? Licenses?

How much does it cost to develop the solution?

# What about maintenance costs?

# How much hardware does the solution require?

Increasing one might make sense if it lowers others…

Any number of criteria we can use for evaluation.

More on that in a bit!

These are participatory events!

Attendees should be taking notes.

Asking questions.

Using their own experiences.

Do you agree? Why or why not?

By the way, you are up next week…

# Pass the briefing filter?

Hands on time.

Workshop it.

Couple of hours.

# A few exercises.

Focus on how to, simple setup.

# Pass the hands on filter?

Time to trial it in the organization.

Real project work that is a good fit.

Probably not a “bet the company” project though!

The new hotness is not our only concern though.

Need to stay current on the things we are using day in day out.

heartbleed.com

# The Heartbleed Bug

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.

## What leaks in practice?

We have tested some of our own services from attacker's perspective. We attacked ourselves from outside, without leaving a trace. Without using any privileged information or credentials we were able steal from ourselves the secret keys used for our X.509 certificates, user names and passwords, instant messages, emails and business critical documents and communication.

## How to stop the leak?

As long as the vulnerable version of OpenSSL is in use it can be abused. Fixed OpenSSL has been released and now it has to be deployed. Operating system vendors and distribution, appliance vendors, independent software vendors have to adopt the fix and notify their users. Service providers and users have to install the fix as it becomes available for the operating systems, networked appliances and software they use.

BUSINESS
SEP 14 2017, 3:21 PM ET

# Equifax Hackers Exploited Months-Old Flaw

by BEN POPKEN

SHARE
Share
Tweet
Email
Print

Equifax announced late Wednesday that the source of the hole in its defenses that enabled hackers to plunder its databases was a massive server bug first revealed in March.

For the rest of the IT world, fixing that flaw was a "hair on fire moment," a security expert said, as companies raced to install patches and secure their servers. But at Equifax, criminals were able to pilfer data from mid-May to July, when the credit bureau says it finally stopped the intrusion.

**Equifax, Software Company Blame Each Other for Security Breach** 1:52

"We know that criminals exploited a U.S. website application vulnerability," Equifax said in an update on its website Wednesday night. "The vulnerability was Apache Struts CVE-2017-5638." Equifax said it was working with a leading cybersecurity firm, reported to be Mandiant, to investigate the breach. Mandiant declined an NBC News request for comment.

**Related:** The One Move to Make After Equifax Breach

The Apache Software Foundation, which oversees the Apache Struts project, said in a press release Thursday that a software update to patch the flaw was

advertisement

FROM THE WEB
Sponsored Links

Find Out In One Minute If You Pre-Qualify For A Citi Card
Citi

Why These 10 SUVs are the Cream of the Crop
Kelley Blue Book

by Taboola

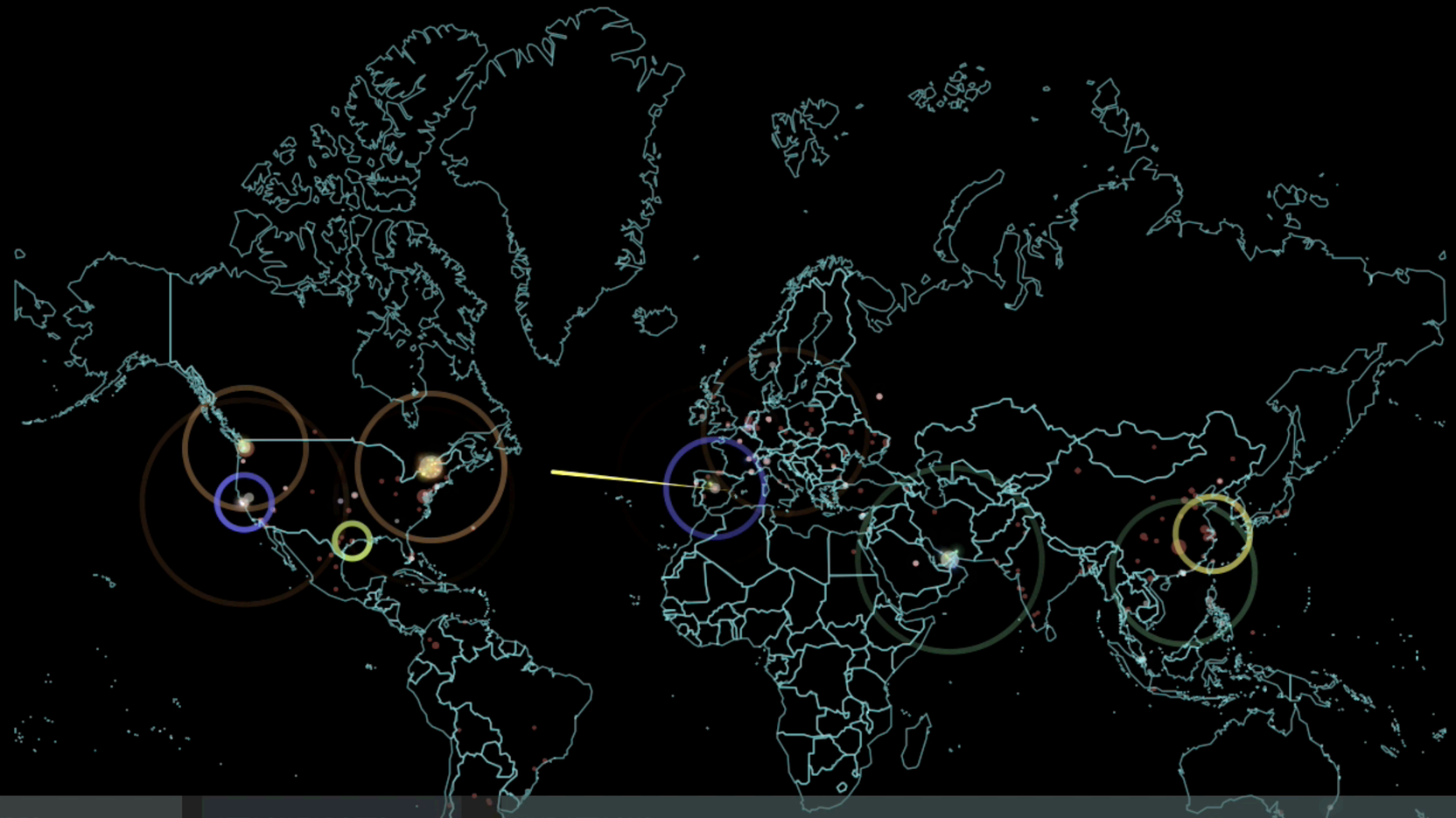
MORE FROM NBC NEWS

Oops.

# Don’t think you’re a target?

map.norsecorp.com

# NORSE



## ATTACK ORIGINS

| # | COUNTRY |
|---|---|
| 525 | United States |
| 381 | China |
| 34 | Ukraine |
| 31 | Netherlands |
| 22 | South Korea |
| 21 | Colombia |
| 18 | Switzerland |
| 14 | France |
| 13 | Vietnam |
| 13 | Turkey |

## ATTACK TYPES

| # | PORT | SERVICE TYPE |
|---|---|---|
| 418 | 25 | smtp |
| 205 | 23 | telnet |
| 192 | 8080 | http-alt |
| 59 | 3389 | ms-wbt-server |
| 41 | 5900 | rfb |
| 35 | 445 | microsoft-ds |
| 31 | 50864 | xsan-filesystem |
| 25 | 3306 | mysql |
| 21 | 53413 | netis-router |
| 18 | 123 | ntp |

## ATTACK TARGETS

| # | COUNTRY |
|---|---|
| 746 | United States |
| 278 | United Arab Emirates |
| 62 | Spain |
| 26 | Singapore |
| 25 | Italy |
| 17 | Philippines |
| 16 | France |
| 11 | Norway |
| 10 | Russia |
| 9 | Belgium |

## LIVE ATTACKS

| TIMESTAMP | ATTACKER | ATTACKER IP | ATTACKER GEO | TARGET GEO | ATTACK TYPE | PORT |
|---|---|---|---|---|---|---|
| 17:39:01.311 | Chinanet Shanghai Province Network | 180.168.28.12 | Shanghai, CN | Montreal, CA | ctdb | 4379 |
| 17:39:01.061 | Telefonica De Espana | 83.52.142.84 | Madrid, ES | San Francisco, US | x11 | 6000 |
| 17:39:00.765 | Microsoft Corporation | 157.56.111.251 | Redmond, US | De Kalb Junctio... | smtp | 25 |
| 17:39:00.414 | An Internet Service Provider In Hong Kong. | 203.132.205.97 | Kowloon, HK | Dubai, AE | ms-wbt-server | 3389 |
| 17:38:59.932 | Xdsl Network-Adsl | 90.178.246.150 | Louny, CZ | San Francisco, US | telnet | 23 |
| 17:38:59.485 | Microsoft Corporation | 65.55.169.248 | Washington, US | De Kalb Junctio... | smtp | 25 |
| 17:38:59.346 | Retecal Sociedad Operadora De Telecomunicacio... | 81.172.32.177 | Leon, ES | De Kalb Junctio... | telnet | 23 |
| 17:38:58.928 | Chinanet Guangxi Province Network | 171.107.91.55 | Nanning, CN | Madrid, ES | telnet | 23 |
| 17:38:58.712 | This Ip Network Is Used For Internet Security Res... | 185.35.62.37 | Geneve, CH | Madrid, ES | ntp | 123 |

HOME

EXPLORE

WHY NORSE?

Justin Smith Follow
Identity and Security Geek
Apr 19, 2016 · 8 min read

# The Three Rs of Enterprise Security: Rotate, Repave, and Repair

At high velocity, the three Rs starve attacks of the resources they need to grow. It's a complete 180-degree change from the traditional careful aversion to change to mitigate risk. Go fast to stay safer—in other words, speed reduces risk.

—Justin Smith

# What is your patching strategy?

What version of X are you on?

Some organizations have a policy of N or N-1.

Do they measure it? Do they enforce it?

What needs to change in your culture to stay at N?

What hurts more? Changing your patching strategy?

Or being on the receiving end of the latest “largest hack ever”?

# What should we evaluate?

Any number of things we can look at.

Litmus tests if you will.

# Fitness functions!

# What is important to you?

# Documentation.

Is there any?

Is it clear?

# Is it up to date?

Is there enough to get you started?

Is there enough to keep you going?

Very different needs in year 2 with a product…

Community.

# How large is it?

# Is it thriving?

# Is it friendly?

# Does it welcome new members?

Or make them feel inferior?

# What does the community value?

Testing? Clean code? Obfuscation?

# Committer diversity.

# Is there a corporate backer?

Do all the committers work at one company?

If so, is it strategic to that company?

# What happens if it isn't strategic tomorrow?

# Codebase.

# Is the code clean?

# Is it readable?

# Are there tests?

Could you fork it if you had to?

# Testability.

# Does it encourage testing?

Or make it nearly impossible?

# Update history.

When was the last commit?

# Is this a dead project?

Or just a mature one?

# Maturity.

# Is it a 0.0.2 product?

Or are we on 3.7?

# Is it changing rapidly?

# Are there breaking changes on the horizon?

# Is there an upgrade path?

How quickly are new versions coming out?

Can your project consume changes at that pace?

**Joe Armstrong**
@joeerl

Dear <redacted>

Thank you for the latest version of your wonderful program with many new features.

Breaking my old code that worked in your previous version is a great idea and will provide me with many hours of fun searching for a work around.

Tearing out my hair
@joeerl

6:28 AM · Nov 15, 2017

**73** Retweets **221** Likes

https://mobile.twitter.com/joeerl/status/930774515512201216

# Stability.

# Are new releases solid?

Or is there a constant stream of patches?

How quickly would you put a new release into prod?

# Extensibility.

# How hard is it to add something?

# Is there a plugin model?

Or are you at the mercy of the committers?

Support.

# Can you buy a support contract?

# Can you find help online?

# Check Stack Overflow…

# Training.

# Can you get training?

# What if you can’t?

Can you build your own?

# Hiring.

# Will you get any hits on a job post?

Does it appear on anyone's resume?

Where does your developers' appetites sit?

Do not underestimate the feelings of your team!

People can take these decisions very personally.

Are your current developers lining up to learn it?

Or avoiding it like the plague?

# Corporate fit.

Does it fit in *your* company?

# What does your company value?

Is it a “breaking change” for your organization?

Will there be any…political roadblocks?

# Licensing.

# How is the product licensed?

# Is it open source?

If so, will your lawyers sign off on the specific license?

Can you company afford to "roll their own" if need be?

If it is commercial, best perform some due diligence on the vendor.

# Are they well funded?

# What is their track record?

Are you comfortable working with them?

# Usage.

Is anyone else using it?

# What are other companies embracing?

Be careful here…

# Comparing Angular and React.

# First things first…

Most of this is highly subjective.

There is no magic formula.

You may want to build something more formal.

# The spreadsheet approach.

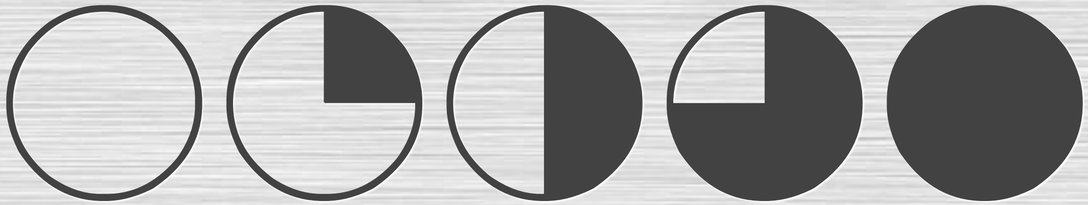
# Options across the top.

Criteria down the left.

Criteria can be weighted.

# Harvey balls.

http://en.wikipedia.org/wiki/Harvey_Balls



How closely does does it map to the criteria?

Very effective...

# What criteria should you use?

How should they be weighted?

That’s up to you!

Second…some people reject the comparison.

Angular is a framework.
React is a library.

Is that too pedantic?

# Documentation.

Plus for both.

Quite extensive.

Both have a good tutorial.

Angular's is a bit more in depth.

Community.

React has a great Code of Conduct for contributors.

But arguably the Angular community is larger.

Angular has been forked more often than React.

But both repositories have similar stars and watches.

Depends a bit on how we treat Angular 1 vs Angular “2”.

# Committer diversity.

Both are backed by large companies.

But contributions are extensive.

More than 1,000 for React and more than 500 for Angular.

React has an extensive page on contributing.

https://facebook.github.io/react/contributing/how-to-contribute.html

Including a set of “good first bugs”!

Angular isn’t quite as strong here.

https://angular.io/contribute

# Codebase.

Both quite clean.

# Testability.

Testability is a central tenant of Angular.

Significant documentation dedicated to testing.

Part of the community.

React is JavaScript centric.

Includes rich set of testing utilities.

Works with whatever testing tools you prefer.

# Update history.

Both are regularly updated.

Arguably Angular 2 was…controversial.

Is that an issue?

Angular 4 and beyond
shouldn't be as difficult.

Angular is committed to a deprecation phase.

And a regular upgrade cadence.

# Maturity.

Angular wins on the maturity argument.

Angular 2 is widely used with 5.2 now available.

React has a higher version number…

And churn should be lower going forward.

# Stability.

Angular updates are very methodical.

Unlikely to see issues
on minor updates.

React makes you stitch together disparate libraries.

Which all move at their own pace. Often rapidly.

Angular has the edge here.

# Extensibility.

Angular works with other libraries.

Free to swap out or modify capabilities.

React lets you make a lot of your own choices.

Very easy to build your own components.

Support.

Both are massive open source projects.

Plenty of search results.

Stack Overflow has *far*
more Angular questions.

Is that good or bad?

# Training.

Training is widely available for both.

Online and in person options are available.

# Hiring.

Angular shows up in more job posts.

But consider who writes most job posts…

Angular has an edge here.

But developers can always learn something new.

# Corporate fit.

I can’t answer this one for you!

Again, both are widely used at a variety of companies.

Is it easier to manage one opinionated library?

Does every version of every library require approval?

Does one already have a toe hold somewhere?

# Usage.

Consider their corporate creators!

Rather large properties.

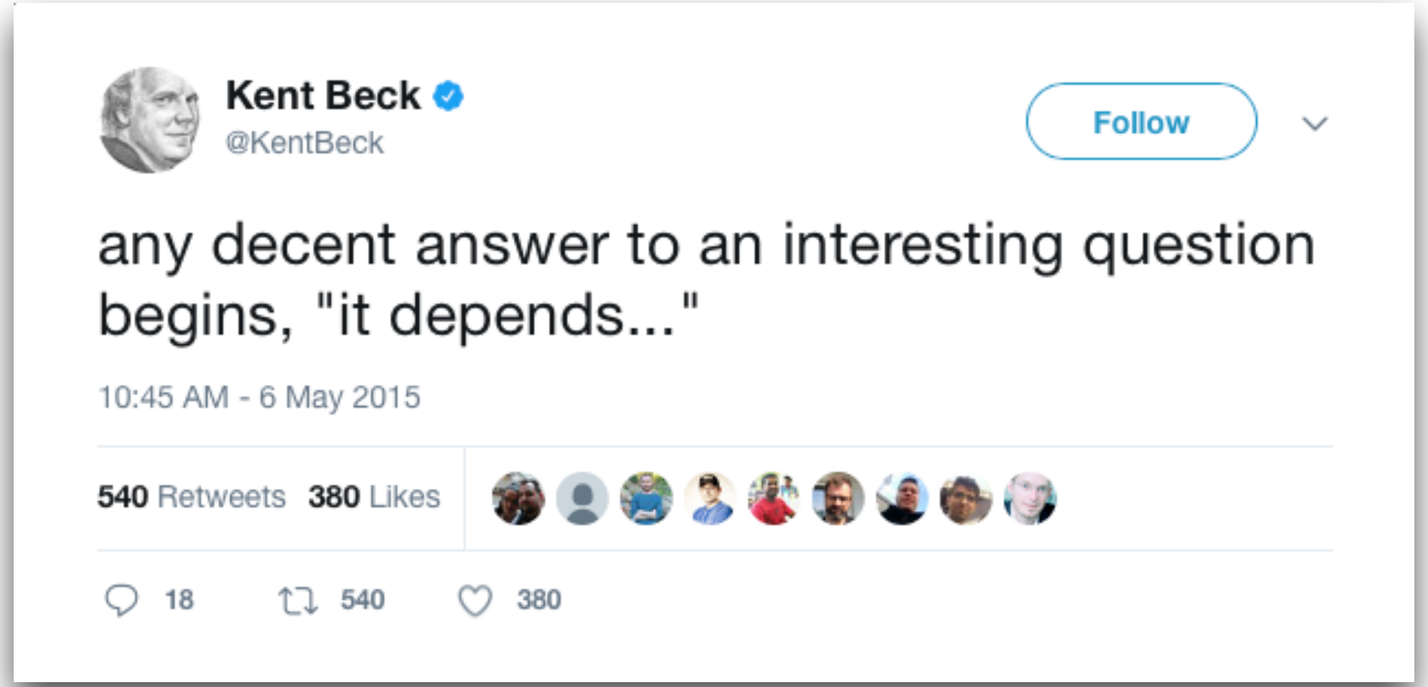
But both frameworks are widely used.

| | Angular | React |
|---|---|---|
| Documentation | ● | ◕ |
| Community | ● | ◕ |
| Committer diversity | ● | ● |
| Codebase | ● | ● |
| Testability | ● | ● |
| Update history | ◔ | ◔ |
| Maturity | ● | ◕ |

| | Angular | React |
|---|---|---|
| Stability | ● | ◕ |
| Extensibility | ● | ● |
| Support | ● | ● |
| Training | ● | ● |
| Hiring | ● | ◕ |
| Corporate fit | ? | ? |
| Usage | ● | ● |

Thoughts? Comments?

Which technology should *you* choose?




https://twitter.com/KentBeck/status/596007846887628801

# What does your project need?

Are you making a corporate level decision?

Or just for your current project?

Do your homework!

Be strategic.

Staying current is an ongoing proposition.

Not a one time activity.

Keep a weather eye out…

# What is coming in the near future?

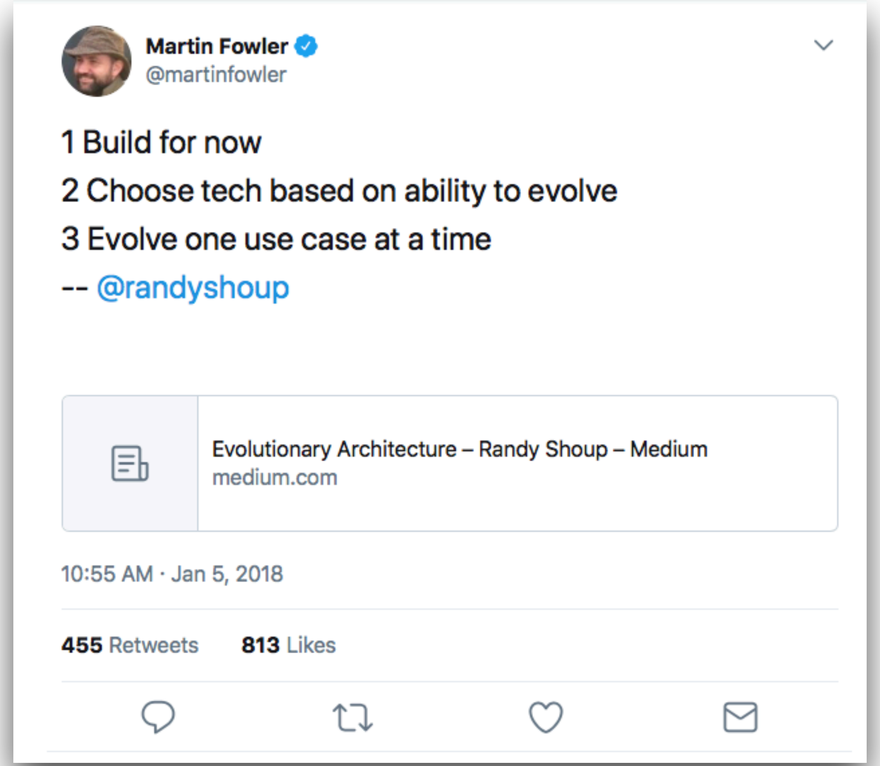
Leverage industry resources.

Use your network.

Get your hands dirty.

Do what is right for *you*!

Consider this guy's advice…




https://mobile.twitter.com/martinfowler/status/949323421619548161

Randy Shoup Follow
Jan 4 · 5 min read

Image Source

# Evolutionary Architecture

At the GOTO 2014 conferences in Copenhagen and Aarhus, I had the opportunity to have an extended set of conversations with Martin Fowler about an idea he had been turning over recently in his head—"Sacrificial Architectures". Every technology architecture is of necessity temporary, and we should both get comfortable with, and take advantage of, that reality. Coincidentally, I'd also been thinking along similar lines—that it is far more a privilege than a burden to get to rewrite a system you have outgrown. So I was inspired to write down a few thoughts about evolutionary architecture.

## Why Are Architectures Temporary?

There is no one perfect architecture for all products and all scales. Any architecture meets a particular set of goals or range of requirements

Never miss a story from **codeburst**, when you sign up for Medium.
Learn more
GET UPDATES

Watch out for analysis paralysis.

Take comfort…

No matter what you choose,
you will be wrong.

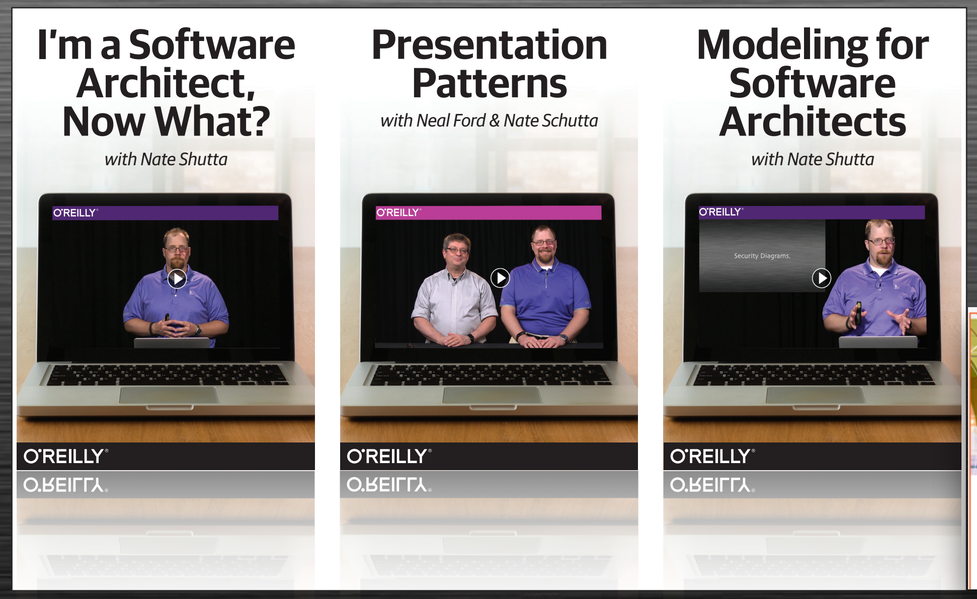
Over a long enough timeframe.

Make decisions based on what we know today.

Good luck!

# Questions?

# Thanks!



Nathaniel T. Schutta
@ntschutta
ntschutta.io