Choosing the web's future

Peter-Paul Koch
http://quirksmode.org
http://twitter.com/ppk

Van Lanschot, 9 February 2017

Four problems

1. Web developers want to emulate native apps, which I think is not possible
2. This causes browser vendors to add more and more features
3. Also, we get more tools that become a problem instead of solving one
4. People who’re new to the web often think the web is just one platform

Emulating native apps

What went before

In 2006-8, several successful web apps were built that emulated native desktop apps; most importantly Google Docs took on Microsoft Office.

Quality was generally good (enough), so this was rightfully seen as a victory for the web.

What came after

After those successes, web developers thought they could do better than native mobile apps as well.

This, generally speaking, has turned out not to be the case

but our feature priorities and the general direction of web development still point toward ever more complicated apps

Not possible

Technically, it’s simple.

Native apps communicate directly with the OS.

Web apps communicate with the browser, which communicates with the OS.

Therefore web apps will always be a bit slower and coarser than native apps.

Not possible

It is impossible for the web to ever become as good as native.

Sure, the web is adopting native feature after native feature, and improving performance by a lot.

It will have caught up with native in … I don’t know, two years?

But by that time native will also have progressed and we’ll still be behind.

Consequences

"You destroy basic usability by hijacking the scrollbar. You take native functionality (scrolling, selection, links, loading) that is fast and efficient and you rewrite it with 'cutting edge' javascript toolkits and frameworks so that it is slow and buggy and broken. You balloon your websites with megabytes of cruft. You ignore best practices. You take something that works and is complementary to your business and turn it into a liability."

Baldur Bjarnason - https://www.baldurbjarnason.com/notes/media-websites-vs-facebook/

Consequences

But wait…

Am I saying that it's all the fault of trying to emulate native apps?

Not quite, though that does play a role. It's the mindset of making everything more complicated that I object to.

Meanwhile, front-enders seem not to understand browsers any more. While understanding browsers is the definition of a front-ender.

2 Too many features

Name all features browsers added in 2016

Name all features browsers added in 2016

You see? You can't

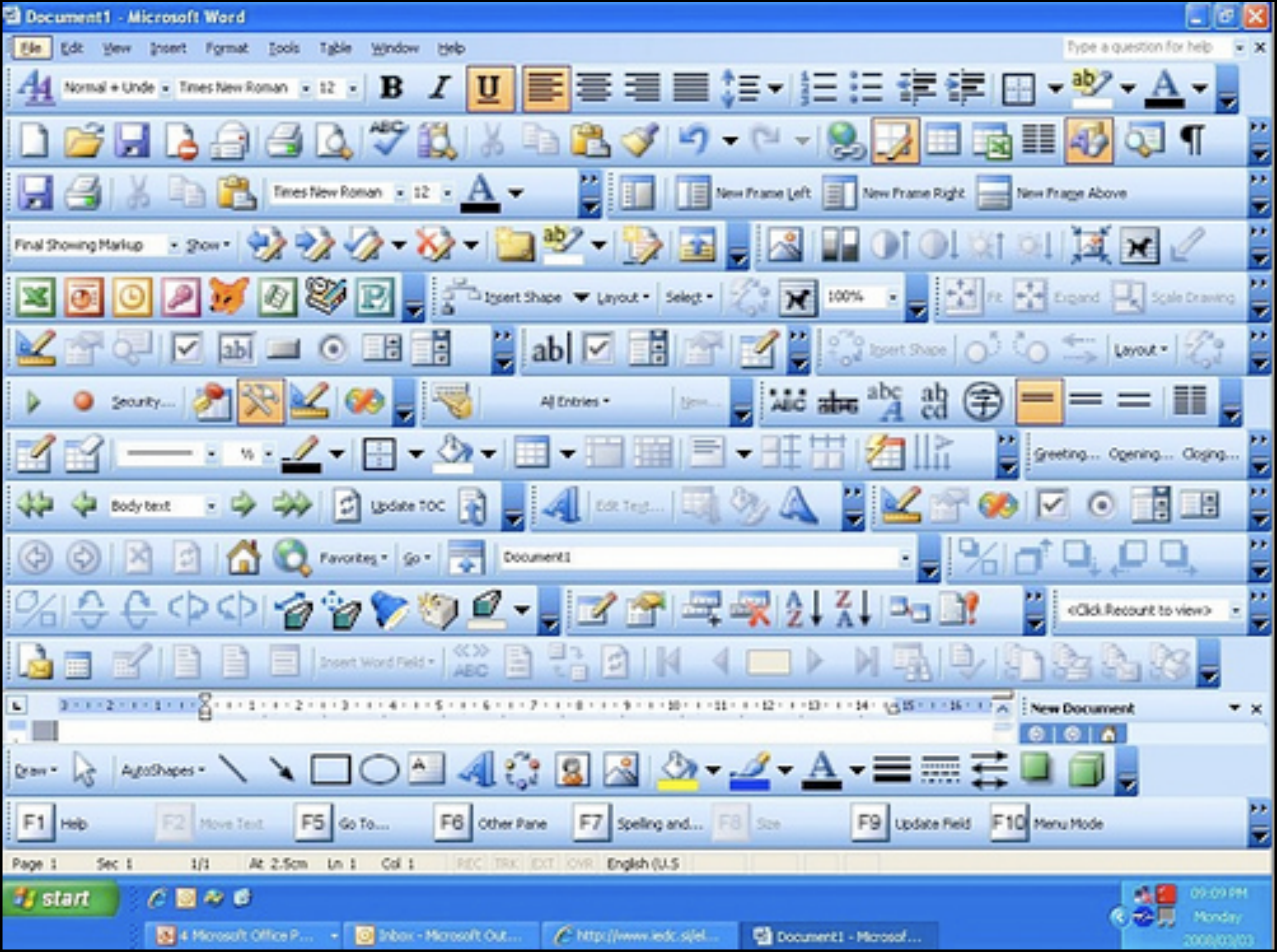
Features

I think browsers are implementing too many features.

This is tricky, though.

It’s not individual features that I object to. Most individual features are a good idea, and they solve some kind of issue.

The problem is that there’s so MANY of them.


Document1 - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Type a question for help
Normal + Unde
Times New Roman
12
New Frame Left
New Frame Right
New Frame Above
Final Showing Markup
Show
Insert Shape
Layout
Select
100%
Fit
Expand
Scale Drawing
Security...
All Entries
New...
Greeting...
Opening...
Closing...
Body text
Update TOC
Edit Text...
Favorites
Go
Document1
<Click Recount to view>
Insert Word Field
New Document
Draw
AutoShapes
F1 Help
F2 Move Text
F5 Go To...
F6 Other Pane
F7 Spelling and...
F8 Size
F9 Update Field
F10 Menu Mode
Page 1
Sec 1
1/1
At 2.5cm
Ln 1
Col 1
REC TRK EXT OVR
English (U.S
start
4 Microsoft Office P...
Inbox - Microsoft Out...
Document1 - Microsof...
09:09 PM
Monday

Polyfills

New features are frequently not supported in many (most?) browsers.

So we add another polyfill. So clever!

Except that it increases our tool footprint once again - possibly even without good cause. Do you REALLY need that new feature?

Also, it makes web developers lazy. Why not force them to write their own? That'll teach them a lot more than just copying code.

Software market maturity

Users = **web developers**, and not visitors!

1. Technology focus. Concentrate on the fact that it works at all.
2. Feature focus. Concentrate on new features users may need.
3. Experience focus. Concentrate on the overall experience users get.

by Jared Spool

Software market maturity

We’ve been stuck in the feature focus phase for far too long.

I’d say it’s time to move to the experience focus stage.

I’d say we want to improve the overall experience of creating websites.

What does that mean? I have no clue.

Moratorium

That’s why I proposed a moratorium on browser features of about a year.

During that year, browsers may not implement new features.

However, browsers are allowed to copy features other browsers already support

and write bug fixes

while developers learn the previous set of features

Stifling innovation

Won’t a moratorium stifle web innovation?

Well yes, it would.

In fact, that’s the point. Since web innovation is currently defined as “making the web even more app-like” it could do with some stifling.

Until we’ve given the whole thing a little more thought.

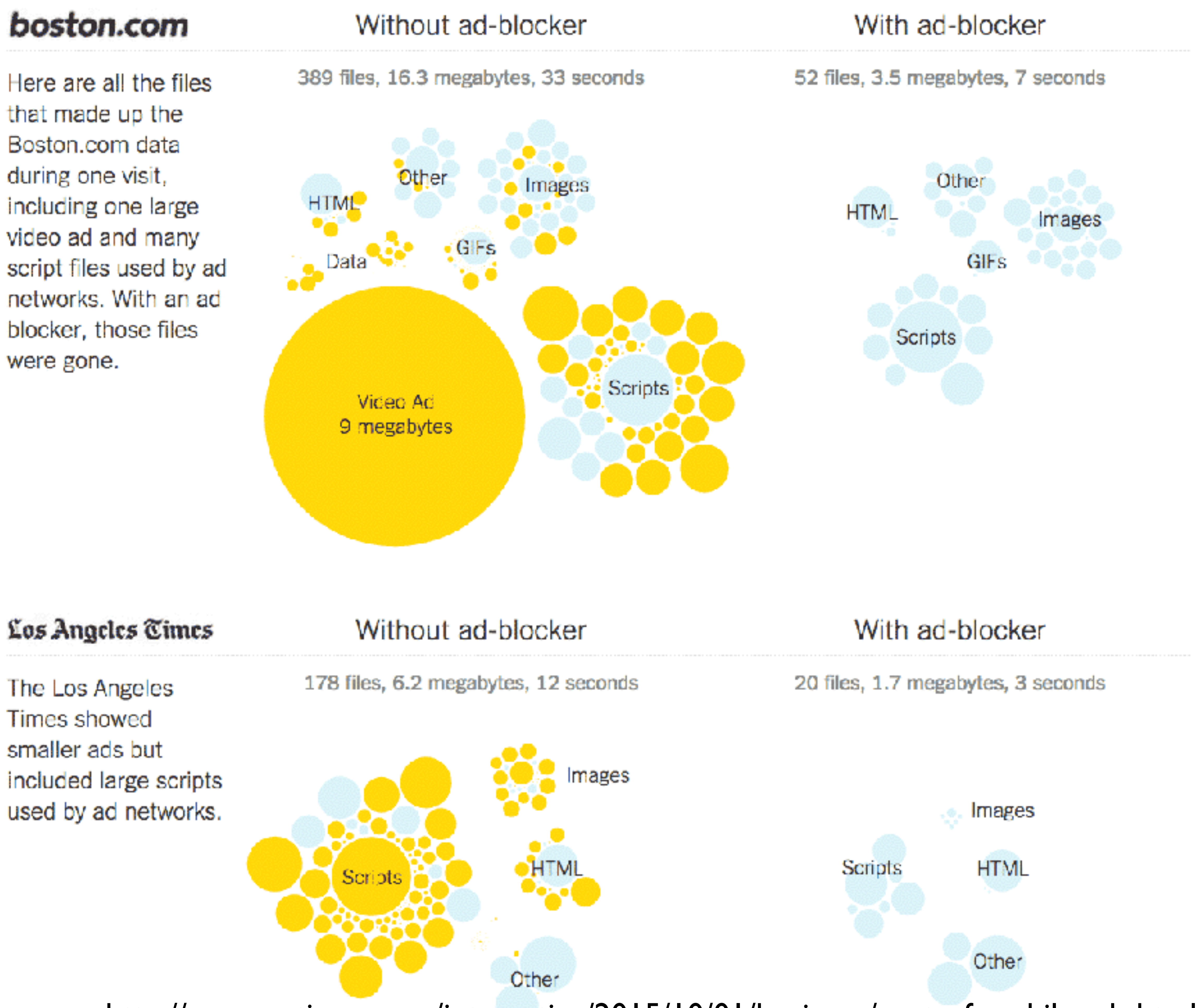
3 Too many tools

Speed

The web has a speed problem, especially on mobile.

Ads are part of the problem. Or rather, maybe not even the ads themselves, but the associated scripts.

The other part of the problem is the tools we use. We're using way too many of them.


boston.com

Without ad-blocker

With ad-blocker

Here are all the files that made up the Boston.com data during one visit, including one large video ad and many script files used by ad networks. With an ad blocker, those files were gone.

389 files, 16.3 megabytes, 33 seconds

52 files, 3.5 megabytes, 7 seconds

Los Angeles Times

Without ad-blocker

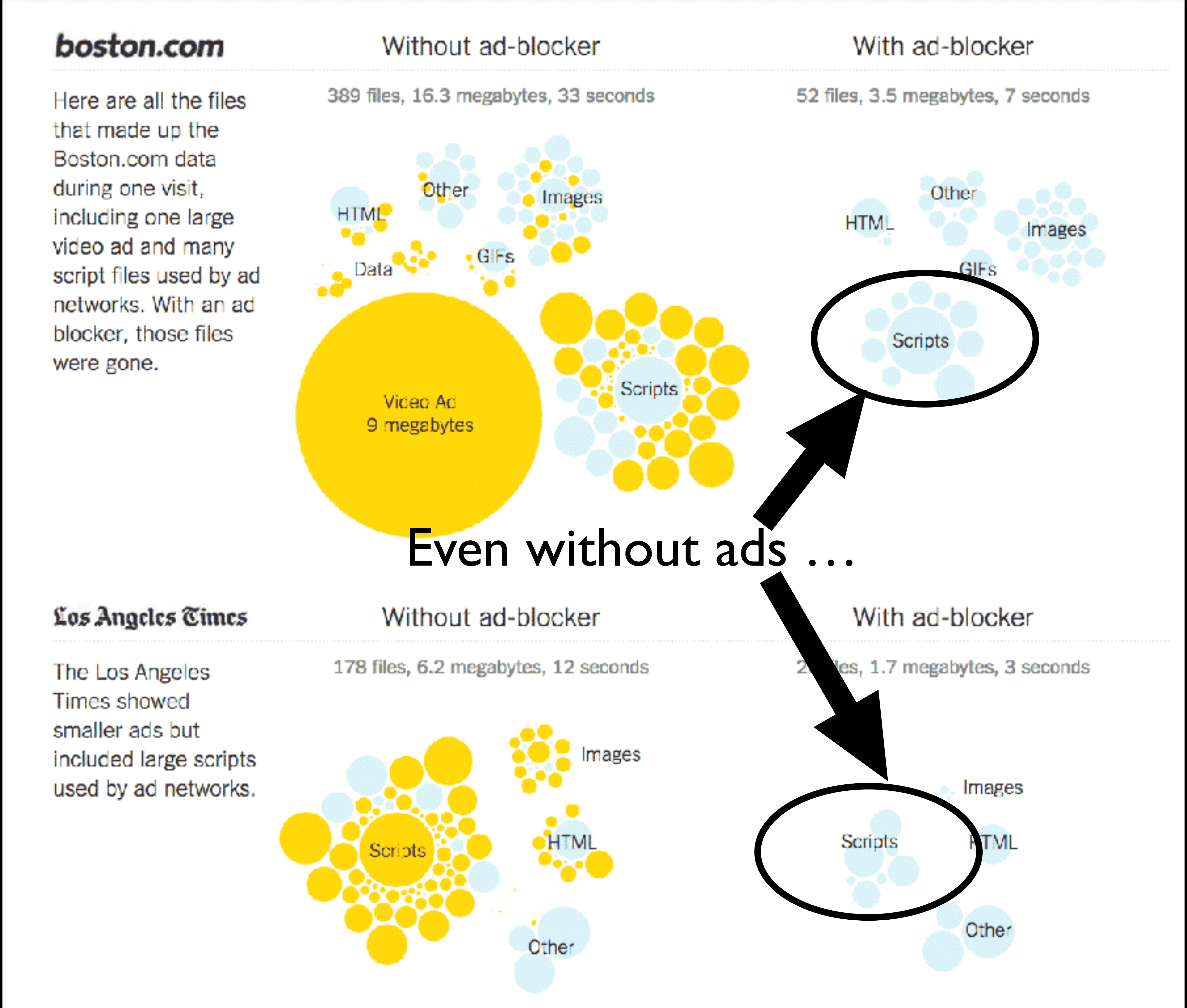
With ad-blocker

The Los Angeles Times showed smaller ads but included large scripts used by ad networks.

178 files, 6.2 megabytes, 12 seconds

20 files, 1.7 megabytes, 3 seconds

http://www.nytimes.com/interactive/2015/10/01/business/cost-of-mobile-ads.html



boston.com

Without ad-blocker

389 files, 16.3 megabytes, 33 seconds

With ad-blocker

52 files, 3.5 megabytes, 7 seconds

Here are all the files that made up the Boston.com data during one visit, including one large video ad and many script files used by ad networks. With an ad blocker, those files were gone.

Even without ads …

Los Angeles Times

Without ad-blocker

178 files, 6.2 megabytes, 12 seconds

With ad-blocker

2[illegible]les, 1.7 megabytes, 3 seconds

The Los Angeles Times showed smaller ads but included large scripts used by ad networks.

Tools

- React
- and React DOM for DOM manipulation
- and Babel for transpiling that cool ES6
- and Browserify for module management
- and Gulp for managing these tasks
- and Typescript for typed variables; and so that you have something to transpile
- yaddah yaddah yaddah

Many thanks to Jose Aguinaga
https://hackernoon.com/how-it-feels-to-learn-javascript-in-2016-d3a717dd577f#.kp42xdqrz

Tools

- React
- and React DOM for DOM manipulation
- and Babel for transpiling that cool ES6
- and Browserify for module management
- and Gulp for managing these tasks
- and Typescript for typed variables; and so that you have something to transpile
- yaddah yaddah yaddah

This is INSANE

We have the best libraries! Our libraries are huuuuge!

Jose Aguinaga

Why so many?

I think we're using this many tools because we want to show web app development is a Serious Thing

and Serious Developers use long toolchains

but these long toolchains run on a server

except on the web, where we force *all of our users* to run them

even when they're on a crappy mobile phone

Anatomy of library use

1. Browser must download the library. This is not a huge problem, since authors have been well aware of this fact for years
2. Then the library must be initialised. This is commonly forgotten

Library initialisation

- Take an Android phone and measure its battery power consumption
- Load a mobile Wikipedia page. It uses jQuery for its show/hide script.
- Then replace jQuery by a five-line custom script.
- 30% power saving.
- 30%! Just for a library you don't need!

http://crypto.stanford.edu/~dabo/papers/browserpower.pdf

Anatomy of library use

1. Browser must download library. This is not a huge problem, since authors have been well aware of this fact for years
2. Then the library must be initialised. This is commonly forgotten

Anatomy of library use

1. Browser must download library. This is not a huge problem, since authors have been well aware of this fact for years
2. Then the library must be initialised. This is commonly forgotten
3. Then the library might need to parse the DOM.

Library DOM parsing

```
<span id="complexId1">{{item.name}}</span>
{{item.value}}<br>{{item.price}}
```

- This sort of code belongs on the server. If a library uses it it's not suited for front-end development.
- Why not? The library has to parse DOM text nodes to find these fragments, and that is the most costly operation imaginable.
- (Looking at you, Angular 1.x)

Library DOM parsing

```
<span id="itemName">Placeholder</span> <span
data-value="itemValue"></span><br><span
class="itemPrice">Placeholder</span>
```

- Instead, libraries should require HTML elements with specific IDs, or other HTML attributes, so that they can be found without parsing the entire DOM.

Anatomy of library use

1. Browser must download library. This is not a huge problem, since authors have been well aware of this fact for years
2. Then the library must be initialised. This is commonly forgotten
3. Then the library might need to parse the DOM.

Anatomy of library use

1. Browser must download library. This is not a huge problem, since authors have been well aware of this fact for years
2. Then the library must be initialised. This is commonly forgotten
3. Then the library might need to parse the DOM
4. Now the library is ready and waits for user input - which might cause costly operations all over again, but we're aware of that

Modularization encourages over-design

John Daggett

Learning

"When in doubt, learn CSS over any sort of tooling around CSS. Learn JavaScript instead of React or Angular or whatever other library seems hot at the moment. Learn HTML. Learn how browsers work.

[…] Focusing on the core helps you to recognize the strengths and limitations of these tools and abstractions. A developer with a solid understanding of vanilla JavaScript can shift fairly easily from React to Angular to Ember."

Tim Kadlec - http://timkadlec.com/2015/09/the-fallacy-of-keeping-up/

The true JavaScripter

- use libraries and frameworks when you need them
- but study them in detail before doing so
- and use a single one per project
- be able to write a medium-complex application without any libraries or frameworks
- which gives you the technical background to change a library or framework if necessary

If you can’t do without tools you’re not a web developer

4 The web platforms (plural)

Browsers are the most hostile

development platforms in the world

Douglas Crockford

Browsers are the most ~~hostile~~ misunderstood development platforms in the world

not quite Douglas Crockford

Web platforms

I feel back-end developers underestimate the web platform, and thus front-end development

because they misunderstand one crucial aspect.

The web is not one platform.

It is a multitude of platforms, most of which you'll never test on.

Platforms

- Why do back-enders expect the web to be one platform?
- They usually work for a manageable number of known environments, where languages, libraries, power and memory, and tools are pre-defined.
- They expect front-end to be just one more of those environment. Once they know JavaScript (i.e. tools) they can work on it.

Explaining the web

Your application

- must run on the five most common Java server environments, all of which bring out a new version every six weeks
- uses four CDNs, two of which have bad APIs
- uses three sets of incompatible libraries, one of which is still in beta
- needs two root certificates that are deliberately incompatible

Explaining the web

- good documentation exists only for two Java server environments and one CDN - the rest is only sketchily documented by other developers
- kernel panics occur *on your users' computers*, and not on your servers
- this entire landscape changes every six months

DRY

- DRY: DO repeat yourself!
- Web development requires you to repeat yourself. If you have an Ajax script that adds data to the page, make sure there's also a simple link somewhere.
- You write the same functionality twice.
- Not all software engineering principles make sense on the web
- because the web is not one platform

Web development

We need to face the fact that software development for the web is different than any other software development.

Problem: web development is not part of computer sciences, but of design.

Because Real Computer Scientists don’t do web. That’s for amateurs.

They refuse to learn about browsers. And without browser knowledge you can’t be a web developer.

5 We’re going wrong

Four problems

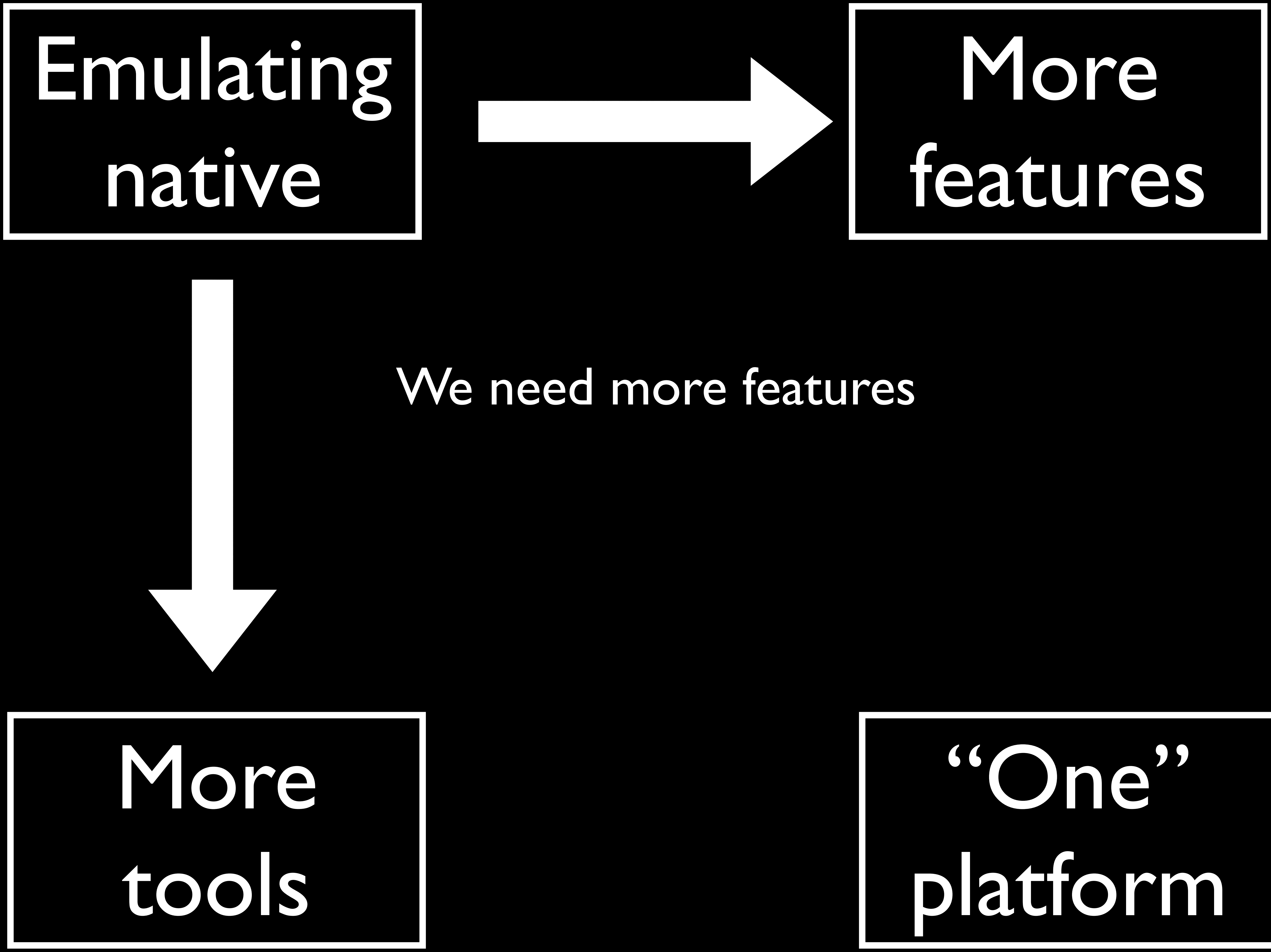
1. Web developers want to emulate native apps, which I think is not possible.
2. This causes browser vendors to add more and more features.
3. Also, we get more tools that become a problem instead of solving one
4. People who’re new to the web often think the web is just one platform

Emulating native

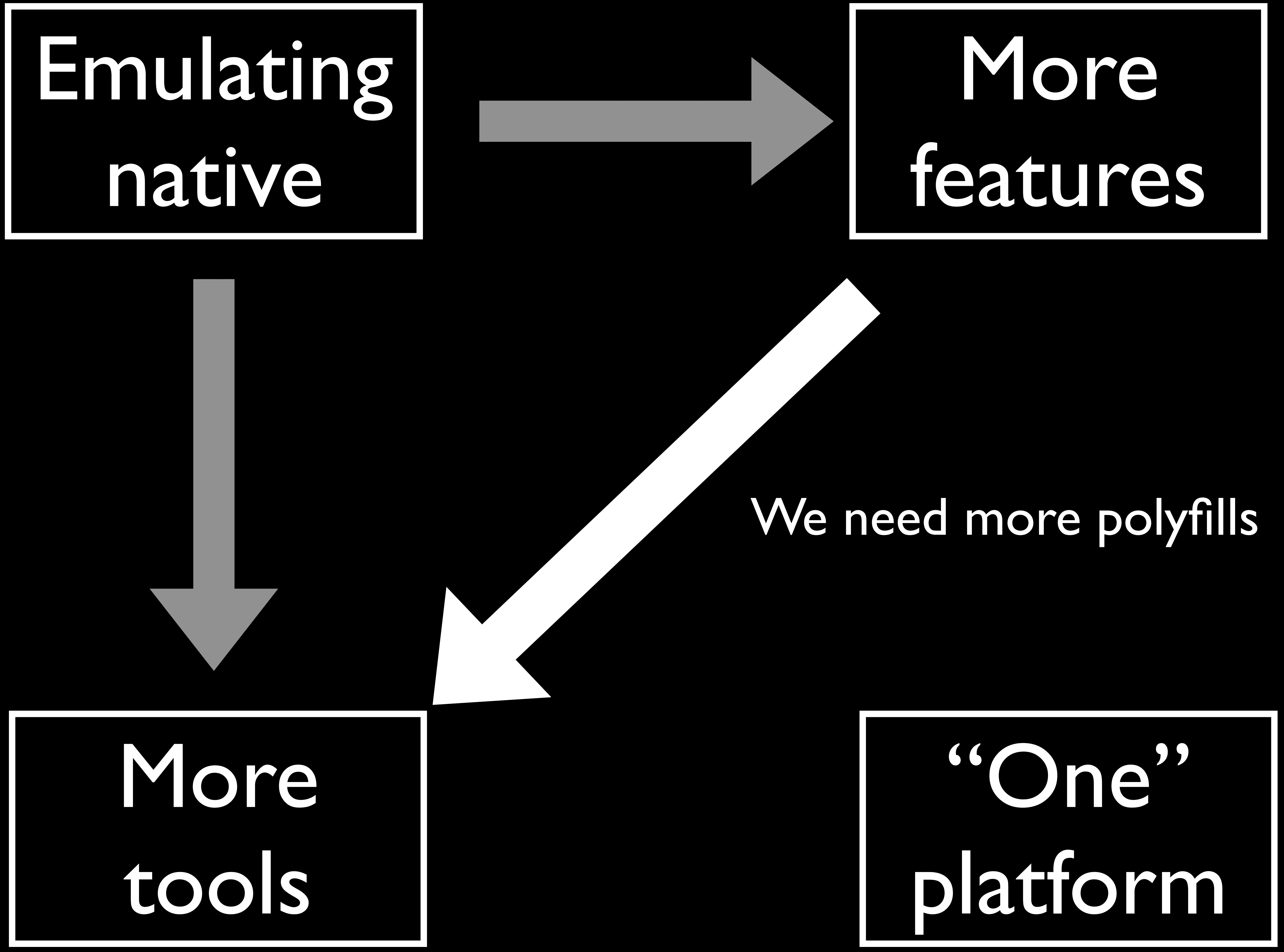
More features

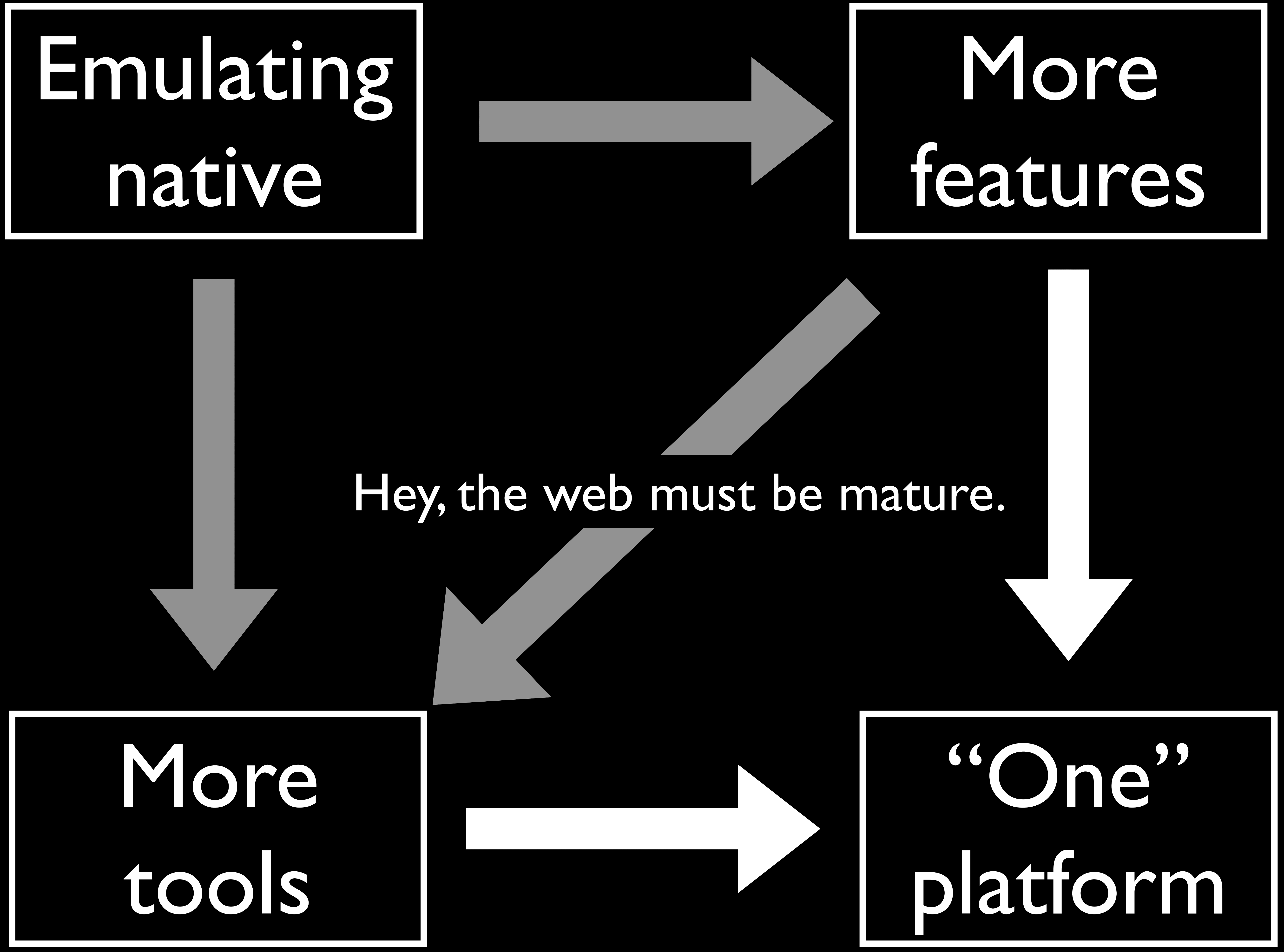
More tools

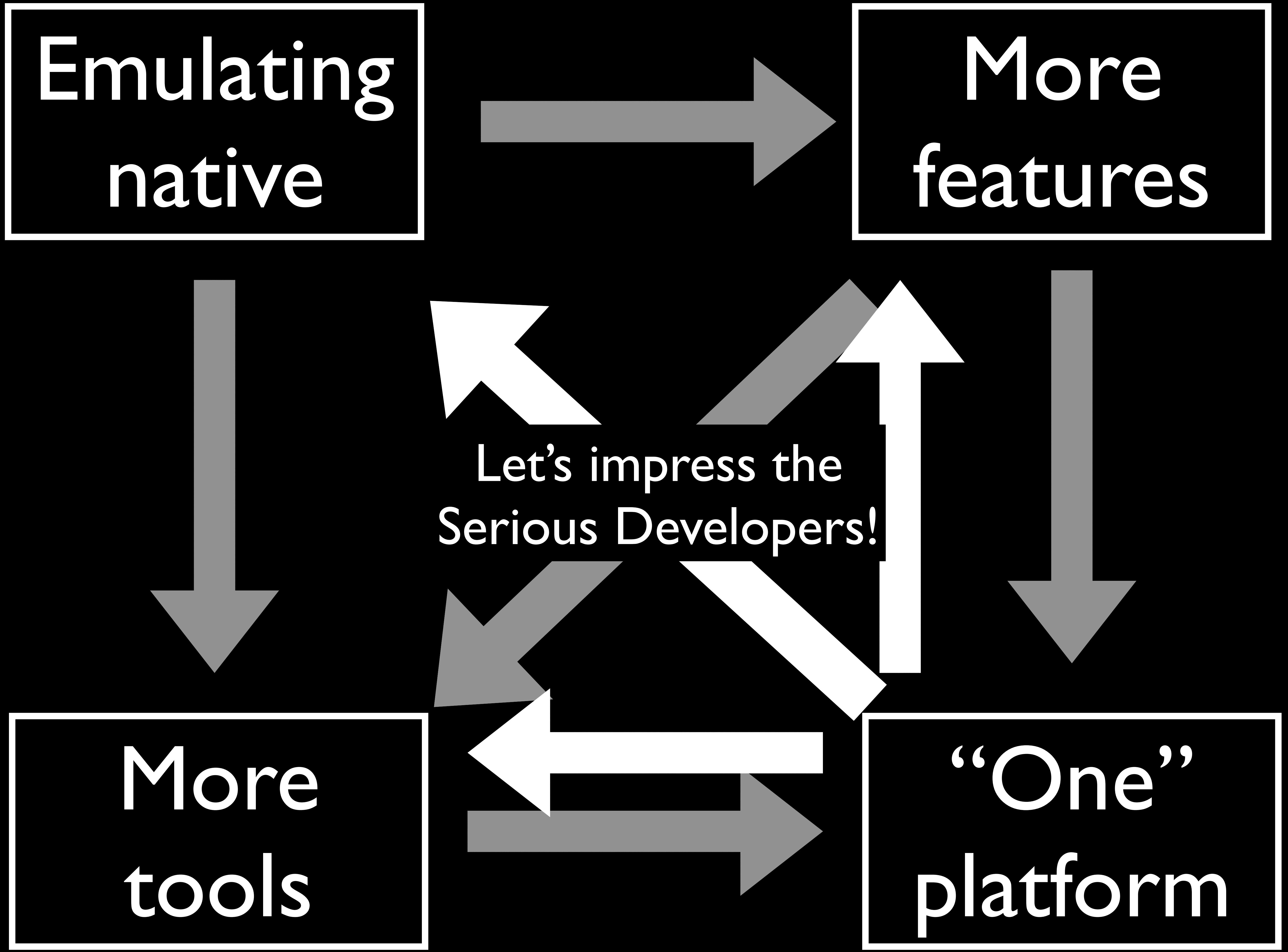
“One” platform



We need more features


Emulating native
More features
We need more polyfills
More tools
"One" platform


Emulating native
More features
Hey, the web must be mature.
More tools
“One” platform


Emulating native
More features
More tools
"One" platform
Let's impress the Serious Developers!

The web’s strengths

Let’s address our featuritis.

That doesn’t mean ditching all tools and features.

Instead, it means thinking about how each individual tool and feature furthers the web’s core strengths:

- URLs
- Reach
- Simplicity

Thank you

I’ll put these slides online

Questions?

Peter-Paul Koch
http://quirksmode.org
http://twitter.com/ppk
Van Lanschot, 9 February 2017