Evolution of the "Web App"

@HenrikJoreteg

@Hoarse_JS

&yet

THIS USED TO BE SIMPLE!

1. WRITE SOME HTML
2. LAY IT OUT WITH FRAMES OR TABLES
3. FTP IT TO A SERVER!
4. BAM!

&yet

CONGRATULATIONS, YOU'RE A WEB DEVELOPER!

&yet

THEN IT GOT HARDER

WHO'S WRITTEN THEIR OWN BLOG SOFTWARE?

&yet

OVER ENGINEERING A BLOG WAS A RITE OF PASSAGE

&yet

1. WRITE SOME PHP/PYTHON/ASP/COLDFUSION
2. SET UP RELATIONAL DATABASE
3. WRITE SOME BAD SQL
4. SHOVE DYNAMIC DATA INTO OUR HTML
5. LAY IT OUT WITH CSS (NO TABLES THIS TIME)
6. RUN IT ON SHARED HOSTING SOMEWHERE

CONGRATULATIONS, YOU'RE A WEB DEVELOPER!

&yet

PHEW!!

&yet

THEN WE GOT "SMART"

&yet

USE A FRAMEWORK!!

&yet

1. RAILS
2. ALL THEM PHP FRAMEWORKS
3. DJANGO
4. ETC.

&yet

OUR EXCESSIVLELY DYNAMIC BLOG...
IS NOW BETTER ORGANIZED
AND HAS MORE DEPENDENCIES

CONGRATULATIONS, YOU'RE A WEB DEVELOPER!

&yet

WHAT ABOUT TODAY?

BACK-END FRONT-END ISOMORPHIC RESPONSIVE ES6(2015) BABEL TRACEUR AMD COMMONJS MVC MVVM FLUX RELAY GULP GRUNT API-GATEWAY CORS JSON-WEB-TOKENS NODE.JS HTTP 2.0 OFFLINE-FIRST MOBILE-FIRST WEBGL WEBRTC WEBSOCKET GRAPHQL AMPERSAND EMBER REALTIME REDIS RIAK LEVELDB RABBITMQ PUSH-NOTIFICATION ENABLED BACKBONE APP WITH POLYFILLED POLYMER WEB COMPONENTS

CONGRATULATIONS YOU MIGHT BE A "WEB DEVELOPER"

&yet

WHAT DOES "WEB DEVELOPER" EVEN MEAN?

WHAT DOES "WEB APP" EVEN MEAN?

THE BROWSER ISN'T OUR RENDERER

&yet

THE BROWSER IS OUR RUNTIME

&yet

THE WEBVIEW IS OUR RUNTIME

&yet

MORE LOGIC MOVING TO FRONT-END JS

&yet

DEVS WITH DIFFERENT BACKGROUNDS CONVERGING ON JS

&yet

BRINGING THEIR PATTERNS AND PREFERENCES

BACK-END FRONT-END ISOMORPHIC RESPONSIVE ES6(2015) BABEL TRACEUR AMD COMMONJS MVC MVVM FLUX RELAY GULP GRUNT API-GATEWAY CORS JSON-WEB-TOKENS NODE.JS HTTP 2.0 OFFLINE-FIRST MOBILE-FIRST WEBGL WEBRTC WEBSOCKET GRAPHQL AMPERSAND EMBER REALTIME REDIS RIAK LEVELDB RABBITMQ PUSH-NOTIFICATION ENABLED BACKBONE APP WITH POLYFILLED POLYMER WEB COMPONENTS

SINGLE PAGE APPS

&yet

HUH?

~~SINGLE PAGE APPS~~

"NATIVE"

I'M A WEB DEVELOPER

NATIVE WEB APP

1. JAVASCRIPT
2. HTML
3. CSS
4. BROWSER APIS

&yet

FUNDAMENTAL DISTINCTION...

THE BROWSER IS YOUR RUNTIME

&yet

SEND THE APP ITSELF
TO THE BROWSER
INSTEAD OF THE
RESULT OF RUNNING IT

```
<!doctype>
<script src="app.1.3.7.js"></script>
```

SHOULD WE EVEN BE DOING THIS?

&yet

SHOULD WE BUILD APPS THAT REQUIRE JAVASCRIPT?

&yet

Engineering Blog

All Twitter Blogs | Search engineering blog

Improving performance on twitter.com

Tuesday, May 29, 2012 | By Twitter (@twitter) [21:23 UTC]

Tweet

To connect you to information in real time, it's important for Twitter to be fast. That's why we've been reviewing our entire technology stack to optimize for speed.

When we shipped #NewTwitter in September 2010, we built it around a web application architecture that pushed all of the UI rendering and logic to JavaScript running on our users' browsers and consumed the Twitter REST API directly, in a similar way to our mobile clients. That architecture broke new ground by offering a number of advantages over a more traditional approach, but it lacked support for various optimizations available only on the server.

To improve the twitter.com experience for everyone, we've been working to take back control of our front-end performance by moving the rendering to the server. This has allowed us to drop our initial page load times to 1/5th of what they were previously and reduce differences in performance across browsers.

On top of the rendered pages, we asynchronously bootstrap a new modular JavaScript application to provide the fully-featured interactive experience our users expect. This

Under the hood

Tools, projects & community

engineering.twitter.com

Tweets Follow

Finagle @finagle 30 Oct

New post by @travisbrown about named entity recognition example project, upcoming improvements to finagle-example: finagle.github.io/blog/2014/10/3...

&yet

SHOULD WE BUILD APPS THAT REQUIRE JAVASCRIPT?

&yet

{{ RAISE YOUR HAND }}

YES!

&yet

WHAT SERVICE ARE WE PROVIDING?

&yet

CONTENT?

CONTENT SHOULD JUST WORK™

&yet

NO REASON TO COMPLICATE THINGS THAT CAN BE SIMPLE

&yet

BUT THE WEB IS
NO LONGER
JUST ABOUT
LINKED CONTENT!

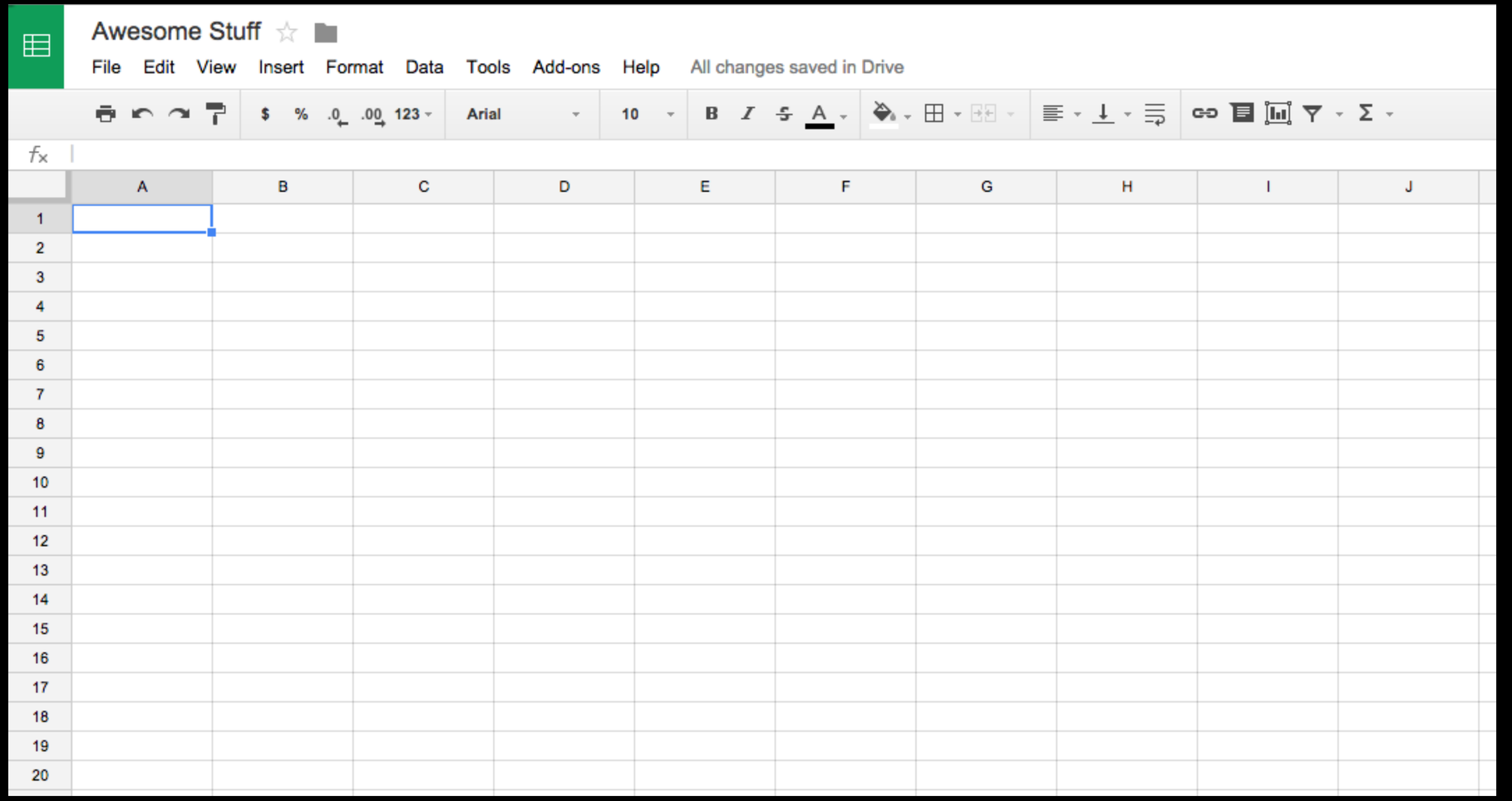
&yet

THERE ARE CASES
WHERE CLIENTSIDE
FUNCTIONALITY
IS THE CORE VALUE
PROVIDED BY SERVICE

&yet


Play on with
Firefox
Mozilla, the non-profit behind Firefox, is bringing high quality gaming to the Web with new technologies like asm.js, which allows native and console games to be played right in your browser. Try the games below and download them to help support our mission to promote openness, innovation and participation on the Internet.
GET THE GAMES
FTL
VOXATRON
DUSTFORCE DX
Humble Bundle
Support
Blog
Log in / Sign up
Humble Bundle
Weekly Bundle
Books Bundle
Mobile Bundle
Humble Store
Humble Origin Bundle 2
Pay What You Want!
Time is running out!
12:21:05:37
$134 worth of awesome games
Pay what you want
Redeem on Origin
Support charity
2 3 7 6 1 3 Bundles sold

&yet


Awesome Stuff
File Edit View Insert Format Data Tools Add-ons Help
All changes saved in Drive
Arial
10
A B C D E F G H I J
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

&yet

&yet


Invite
Lock
Leave
Your Name
Mute
Hide
Share screen
show all / current speaker
anonymous
Jenn
Help · Feedback
anonymous
Jenn
Chat

&yet

HIGH PERFORMANCE

1. RENDERING
2. NETWORKING
3. FILE READ/WRITE
4. STORAGE
5. WEB AUDIO APIS
6. WEBGL
7. VOICE/VIDEO

&yet

BROWSERS ARE NOT
DUMB DOCUMENT VIEWERS

&yet

MOST CAPABLE
UBIQUITOUS
RUNTIMES
ON THE PLANET

&yet

I'M JUST GOING TO SAY IT:

THERE ARE TWO TYPES OF APPLICATIONS ON THE WEB

&yet

1. NATIVE WEB APPS

2. SERVER-SIDE WEB APPS

THEY ARE
FUNDAMENTALLY
DIFFERENT

&yet

AND THAT'S O.K.

ANYTHING
WE CAN BUILD
WITH WEB TECH
I THINK WE SHOULD

&yet

EVEN IF WE CAN'T SUPPORT OLDER BROWSERS

&yet

THE WEB IS INFINITELY MORE
OPEN
THAN NATIVE PLATFORMS

&yet

USER EXPECTATIONS HAVE EVOLVED

THE WEB IS DOING PRETTY WELL ON DESKTOPS

&yet

THE WEB IS LOSING ON MOBILE

THE WEB IS LOSING ON EXPERIENCE

WE OFTEN PREFER NATIVE APPS TO THE WEB

&yet

QUALITY AND POLISH
OF USER EXPERIENCE
IS OFTEN MUCH BETTER

&yet

LET'S FIX THIS!

WE'RE TOO FOCUSED
ON THE PAST INSTEAD OF
COMPETING ON EXPERIENCE

&yet

SAYING THERE'S A DISTINCTION
MAKES SOME PEOPLE MAD

&yet

"Everything should be an enhancement!"

WE'RE ON THE SAME TEAM!

&yet

WE WANT THE OPEN WEB TO WIN!

&yet

HOW COULD WE EVEN BUILD A PROGRESSIVELY ENHANCED VERSION OF TALKY?

&yet

SHOULD WE NOT HAVE BUILT IT?

WHERE'S THE DOWNSIDE?

LET'S LOOK A BIT CLOSER AT TWITTER

&yet

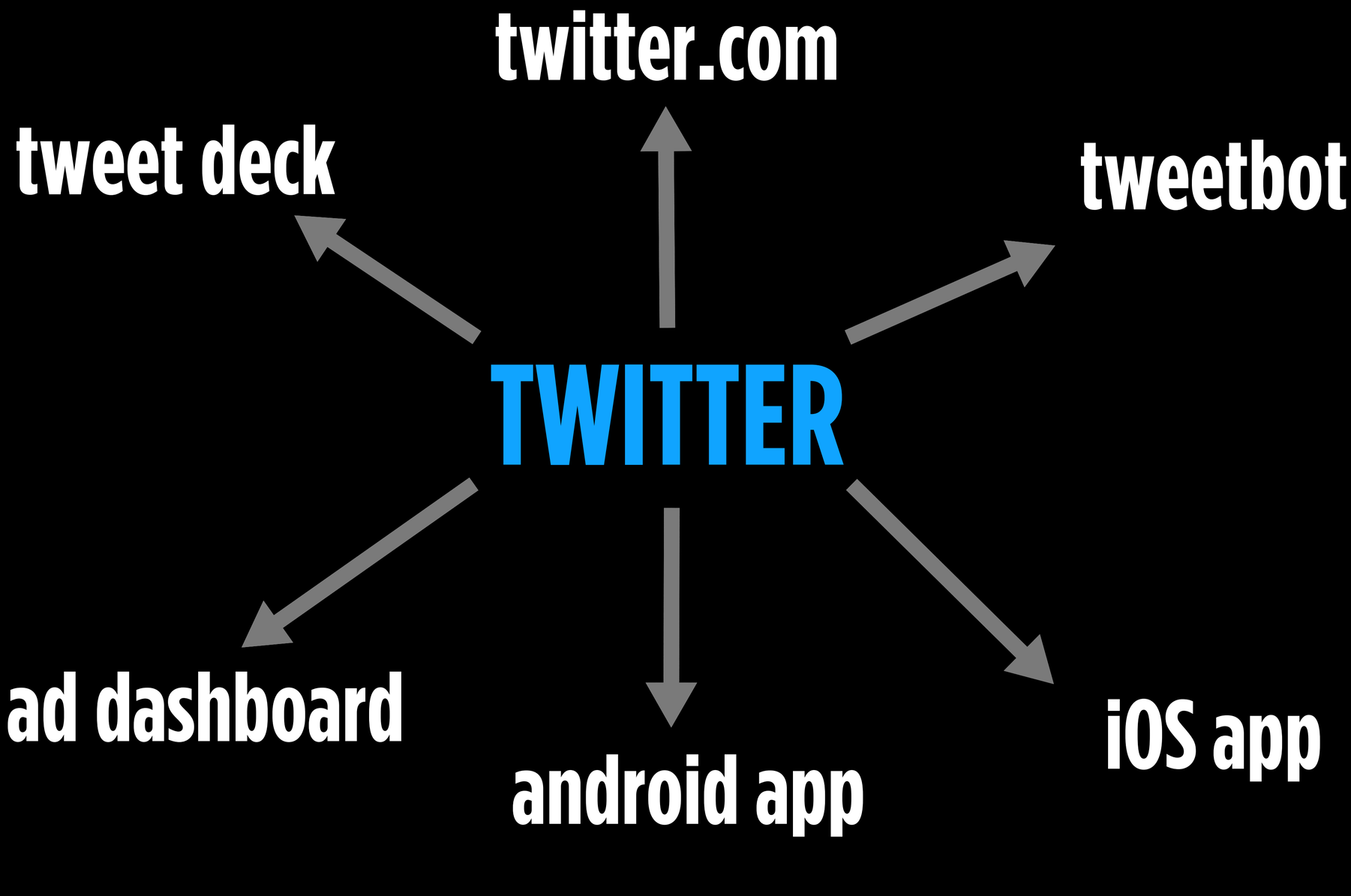
WHAT IS TWITTER?

IS IT A WEB APP?

NO.

IT'S A SERVICE

APP != SERVICE


twitter.com
tweet deck
tweetbot
TWITTER
ad dashboard
android app
iOS app

&yet

I DIDN’T REALLY
CARE HOW THEY
BUILT THEIR WEBAPP

&yet

BECAUSE I DIDN'T USE IT ANYWAY!

I WAS USING AN iOS APP!

THEIR WEB APP HAD
ALREADY FAILED ME!

LET'S THINK ABOUT THIS

WHEN I FOLLOW
A LINK TO A
RANDOM TWEET
ON MY PHONE...

JUST LET ME READ IT!

I DON'T MIND IF IT'S

plain text

DON'T MAKE ME
DOWNLOAD
2MB OF JS
TO READ
140 CHARACTERS
OF TEXT!

&yet

THIS IS THE PROBLEM
THEY FIXED WITH
NEW NEW TWITTER

&yet

BUT...

CATCHING UP WITH ALL THINGS TWITTER IS A FUNDAMENTALLY DIFFERENT USE CASE

&yet

FAILING TO RECOGNIZE DISTINCTION MAKES US FLOUNDER

&yet

A SERVICE CAN PROVIDE BOTH!

&yet

TWITTER.COM
&
TWEETDECK.COM

THERE'S STILL SOME GAPS BETWEEN WEB AND NATIVE

&yet

REAL OFFLINE SUPPORT

PLATFORMS THAT TREAT
NATIVE WEB APPS AS
FIRST CLASS
CITIZENS

&yet

THOSE THINGS ARE CHANGING

1. SERVICE WORKER

1. SERVICE WORKER

PROGRAMMABLE CACHE LAYER

CAN INTERCEPTS ALL NETWORK REQUESTS

THIS IS HUGE!

&yet

2. INSTALLABLE WEB APPS

JSON-BASED WEB MANIFEST

CHROME M39+
FIREFOX

https://developer.chrome.com/multidevice/android/installtohomescreen

https://w3c.github.io/manifest/

1. SIGNAL INTENT
2. UNINSTALL
3. DEEPER DEVICE APIS

"What about performance?"

WHITE PAGE OF DEATH

TIME TO FIRST PAINT

A PRIMED CACHE LARGELY INVALIDATES THIS ARGUMENT

&yet

1. GIVE IT A UNIQUE NAME

```
<!doctype>
<script src="app-1.2.7.js"></script>
```

2. CACHE IT FOREVER

```
HTTP/1.1 200 OK
Cache-Control: max-age=REALLY BIG NUMBER!
Content-Encoding: gzip
```

&yet

MOST OF THESE
TYPES OF APPS
REQUIRE YOU
TO BE LOGGED IN

&yet

PRE-FETCH APP ON PUBLIC PAGES OR LOGIN PAGE

&yet

NATIVE WEB APPS CAN STILL HAVE SMALL JS PAYLOADS!

&yet

ALL JS IN THE AMPERSAND.JS APP ON TODOMVC.COM

COMBINED

28kb min + gzip

&yet

SMALLER THAN JQUERY 2.0

THE OTHER ASPECT OF PERFORMANCE...

ONCE LOADED, PERFORMANCE IS WAY BETTER!

&yet

IF I'M GOING TO
LEAVE APP OPEN
ON MY DESKTOP
I CARE **WAY LESS**
ABOUT LOAD TIME

&yet

"What about dual rendered
a.k.a. isomorphic apps?"

&yet

JUST A CLIENTSIDE APP WITH AN OPTIMIZED INITIAL RENDER

GOING FULL-ISOMORPHIC RENDERING, WITH USER DATA AND ALL...

&yet

OFTEN REQUIRES
DRAMATICALLY
MORE COMPLEX CODE

THERE ARE SOME CASES WHERE IT MAKES SENSE

&yet

WITH STATE OF TOOLING TODAY OFTEN NOT WORTH THE COMPLEXITY

&yet

HOWEVER...

DOESN’T HAVE TO BE ALL OR NOTHING

WE CAN PRE-RENDER EVERYTHING THAT'S NOT USER-SPECIFIC DATA

&yet

WE CAN DO THIS AS
A FULLY STATIC SITE

&yet

Route:	Asset:
site.com/pic.png	-> pic.png
site.com	-> index.html
site.com/page	-> page.html
site.com/asdf	-> 404.html or 200.html

&yet

WOAH!

APPS USUALLY HAVE:

1. public/marketing pages
2. all the stuff behind the login

&yet

WRITE "PUBLIC" PAGES AND APP LAYOUT HTML AS ISOMORPHIC COMPONENTS

PRE-RENDER THEM TO STATIC PAGES AT BUILD TIME!

&yet

SLIP IN THE BUILT JS BEFORE:
</body>

index.html: public home page
200.html: application layout

COULD POTENTIALLY EVEN DO THIS FOR DYNAMIC/PUBLIC DATA

THINK ABOUT WHAT WE GET

pixels on the screen immediately

TOTALLY CRAWLABLE (SEO)

JS TAKES OVER ROUTING WHEN LOADED

DEPLOYMENT AND OPS BECOME AS SIMPLE AS FTP

WRITE 1 VERSION OF YOUR APP
GET 90% OF BENEFIT FROM ISOMORPHIC RENDERING

&yet

USERS WILL END UP WITH A PRIMED CACHE JUST BY VISITING YOUR MARKETING PAGES

&yet

READY FOR: PHONEGAP/CORDOVA DESKTOP APP

&yet

NOW WE HAVE AN APP WITH A SINGULAR CONCERN: PRESENTATION

&yet

I'VE STARTING BUILDING ALL MY APPS AS STATIC NATIVE WEB APPS

&yet

TOTALLY <3 IT!

FOR SO LONG THE TREND
HAS BEEN TOWARD COMPLEXITY


Henrik Joreteg
@HenrikJoreteg
If you don't actively fight for simplicity in software, complexity will win.
…and it will suck.
RETWEETS
515
FAVORITES
278
11:00 PM - 6 Aug 2013

&yet

WHAT'S THE NEXT STEP IN THE EVOLUTION OF THE "WEB APP"?

&yet

GOING BACK TO SIMPLE

&yet

GOING BACK TO THE STATIC WEB

&yet

STATIC NATIVE WEB APPS

POWERED BY SERVICES
SOME WHICH WE BUILD
MANY OF WHICH WE RENT

&yet

surge.sh
hood.ie
firebase.com
auth0.com
divshot.com

SIMPLE OPEN SOURCE EXAMPLE:
HubTags.com

&yet

- React
- Ampersand.js
- Webpack (hjs-webpack)
- GitHub API
- Surge.sh

&yet

andyet.com

&yet

HOW CAN WE BE SURE WE'RE BUILDING WITH THE RIGHT TOOLS?!

&yet

WE CAN'T!

WHAT DO WE KNOW?

THINGS WILL CHANGE

&yet

BUILD MODULAR SYSTEMS THAT STRIVE TO BE AS SIMPLE AS THEY CAN BE

OFFLOADING PRESENTATION
STATIC NATIVE WEB APPS

BUILDING MICROSERVICES TO ENABLE THAT TYPE OF APP

OPTIMIZE FOR CHANGE.
IT IS THE ONLY CONSTANT.

&yet

LET'S KEEP PUSHING FOR SIMPLICITY

LET'S BUILD FOR THE FUTURE OF THE WEB, NOT ITS PAST

&yet

THANKS!

@HenrikJoreteg, andyet.com

&yet