COPENHAGEN
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2014

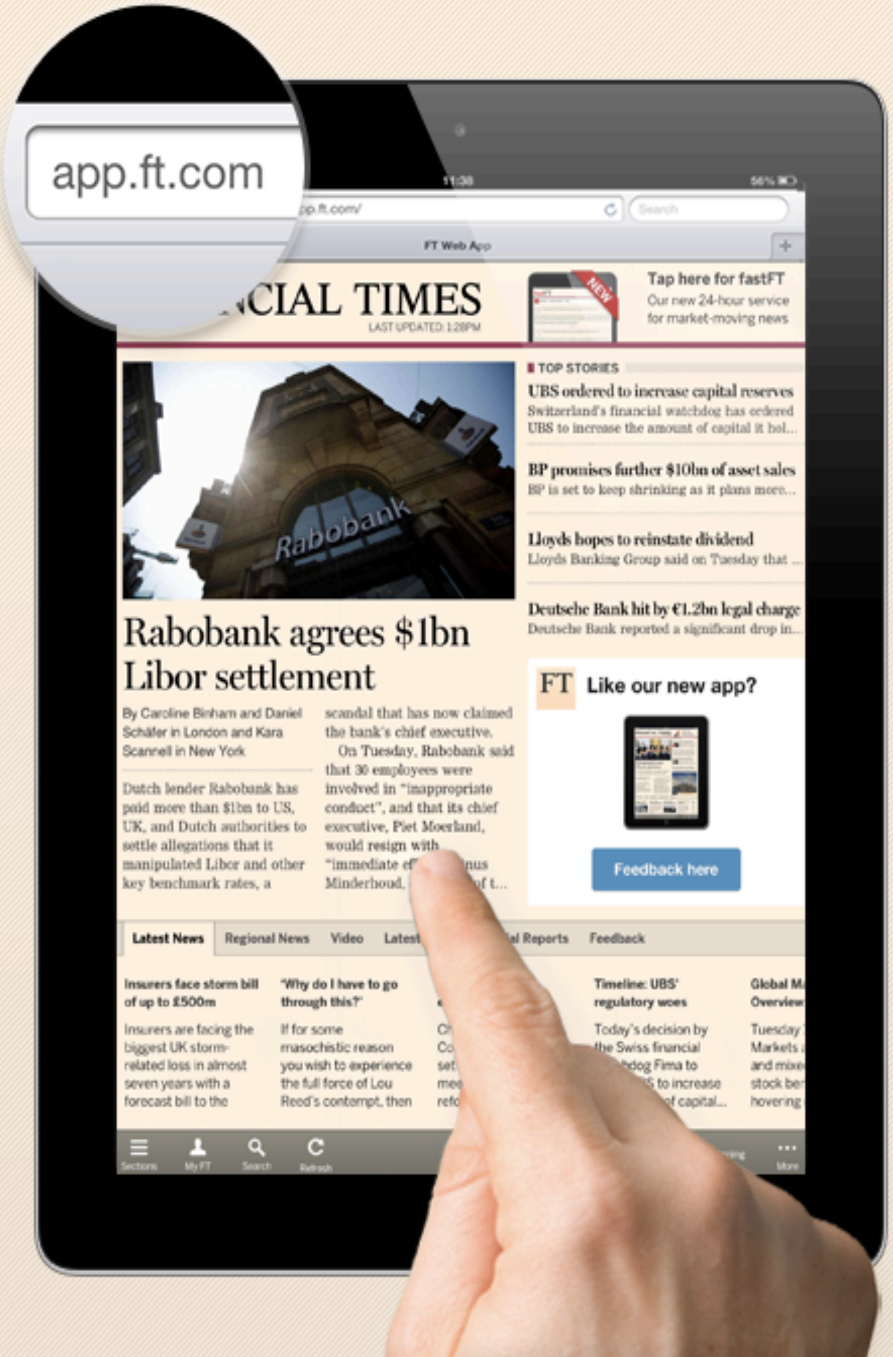
goto; conference

CHALLENGES OF LARGE WEB APPS

Graham Hinchly
Engineering Manager, FT Labs

app.ft.com

- Only mobile optimised experience for FT content
- Lots and lots of JS & CSS
- Works offline
- Momentum scrolling & swiping



UBS ordered to increase capital reserves
Switzerland's financial watchdog has ordered UBS to increase the amount of capital it hol...

BP promises further $10bn of asset sales
BP is set to keep shrinking as it plans more...

Lloyds hopes to reinstate dividend
Lloyds Banking Group said on Tuesday that ...

Deutsche Bank hit by €1.2bn legal charge
Deutsche Bank reported a significant drop in...

Rabobank agrees $1bn Libor settlement

By Caroline Binham and Daniel Schäfer in London and Kara Scannell in New York

Dutch lender Rabobank has paid more than $1bn to US, UK, and Dutch authorities to settle allegations that it manipulated Libor and other key benchmark rates, a scandal that has now claimed the bank's chief executive.

On Tuesday, Rabobank said that 30 employees were involved in "inappropriate conduct", and that its chief executive, Piet Moerland, would resign with "immediate ef[illegible]nus Minderhoud, [illegible]f t...

Insurers face storm bill of up to £500m
Insurers are facing the biggest UK storm-related loss in almost seven years with a forecast bill to the

'Why do I have to go through this?'
If for some masochistic reason you wish to experience the full force of Lou Reed's contempt, then

Timeline: UBS' regulatory woes
Today's decision by the Swiss financial [illegible]dog Fima to [illegible]S to increase [illegible]of capital...

Global M[illegible] Overview
Tuesday[illegible] Markets [illegible] and mixe[illegible] stock be[illegible] hovering

What Am I Going to Talk About?

- Web app architecture (mostly JS)
- UI Performance
- Offline

Architecture

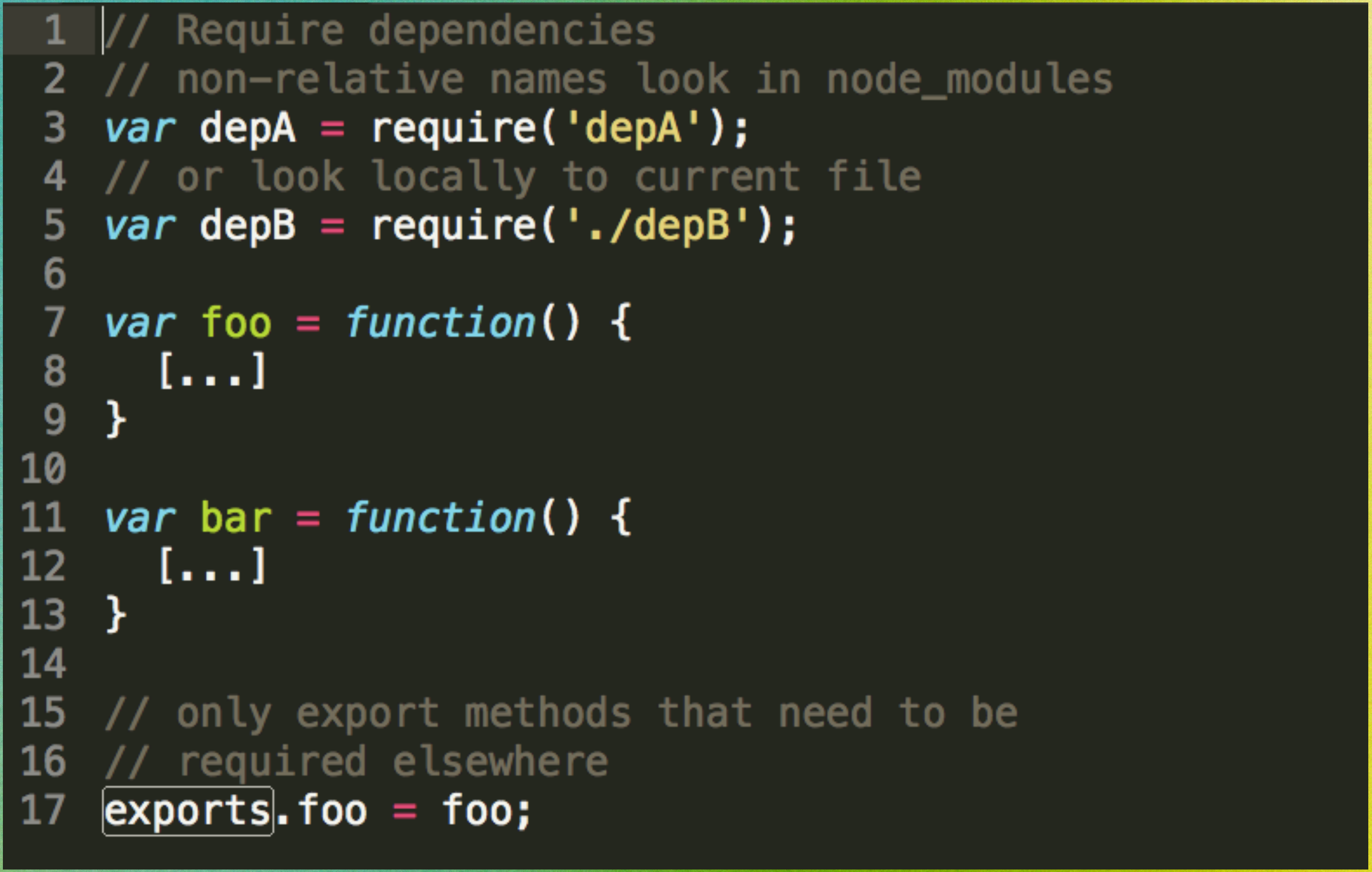
What are we aiming for?

- Safe to extend/modify
  – Clear principles
- Reusability
  – Independent of context
- Happy team ☺

Modular Javascript

- We want
  - Encapsulation
  - Dependency management
  - Reusability on client and server

Browserify



```
// Require dependencies
// non-relative names look in node_modules
var depA = require('depA');
// or look locally to current file
var depB = require('./depB');

var foo = function() {
  [...]
}

var bar = function() {
  [...]
}

// only export methods that need to be
// required elsewhere
exports.foo = foo;
```

Looks Great, but…

- Difficult to apply to legacy code, beware circular dependencies
  - Prefer refactoring over workarounds
- Can be slow with large codebase - needs to be re-run on each change

Dependency Management

- npm
  - Great for pure Javascript
  - Ensuring repeatability
    - Tags and/or semver can be unreliable
    - Want to pin exact commit, so use npm-shrinkwrap
  - Reliance on registry as part of build/ deployment (inc. for CI) can be problematic
    - We use a private lazy cache

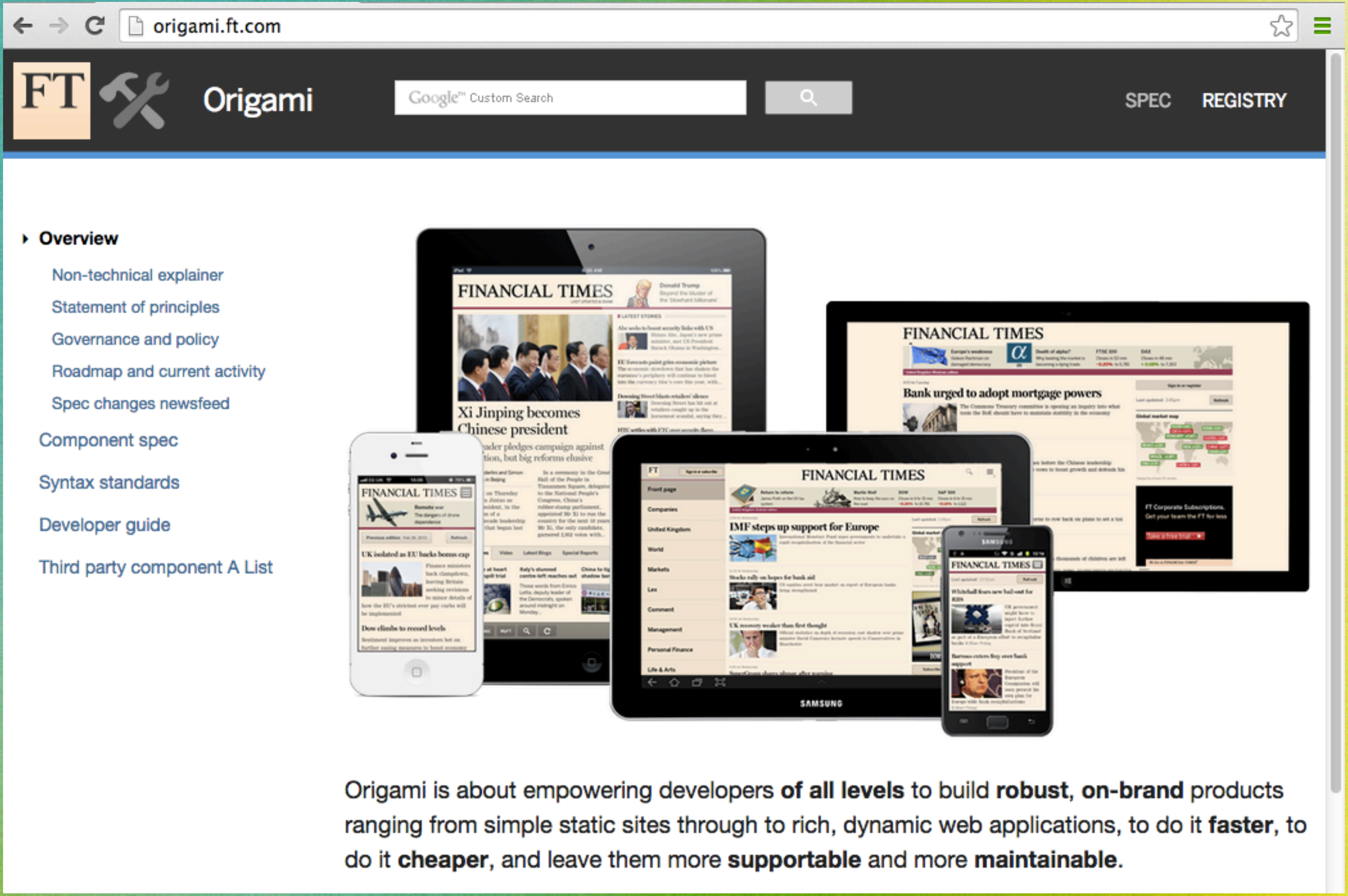
Breaking Up a Monolith

- We can extract self contained modules out of code base by using npm + browserify
  - And start to write JS unit tests
- Can then just `npm install … --save`
- And `require(‘…’)` in our code

Beyond One App…

- But what about sharing entire components between multiple sites/apps?
- Components require mix of HTML, CSS & JS
  - Seems to currently be best served by **Bower**
- **Origami:** open initiative at the FT to promote shared components

origami.ft.com



- **Overview**
  - Non-technical explainer
  - Statement of principles
  - Governance and policy
  - Roadmap and current activity
  - Spec changes newsfeed
- Component spec
- Syntax standards
- Developer guide
- Third party component A List

Origami is about empowering developers **of all levels** to build **robust, on-brand** products ranging from simple static sites through to rich, dynamic web applications, to do it **faster**, to do it **cheaper**, and leave them more **supportable** and more **maintainable**.

The Future…?

- What about sharing with the entire web?
  - Web components
    - Maintain semantic, declarative HTML
    - Import components from other authors
    - CSS & DOM encapsulation via ShadowDOM
    - Latest (updated 9th Sept) css-tricks.com/modular-future-web-components/
- How can we pull in component at runtime but still store it for offline use?

Performance

What are we aiming for?

- Smoothness
- Responsiveness
- Crash-free

Why Is This hard?

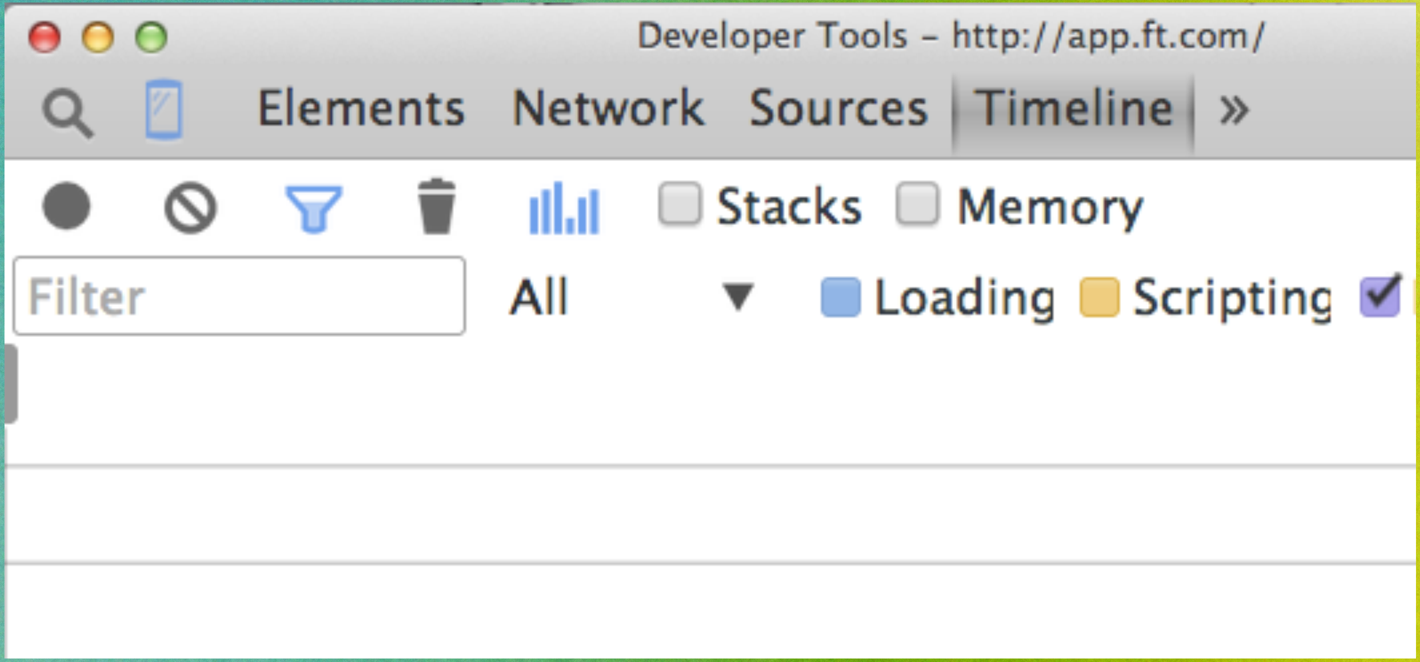
- Layout was designed for documents
  - This is great as we didn’t need to worry about positioning everything exactly
  - But this means small changes can affect lots of things
  - Browser needs to do lots of recalculation
- Javascript is single threaded – blocks UI
- “Automatic” memory management

Achieving Smoothness

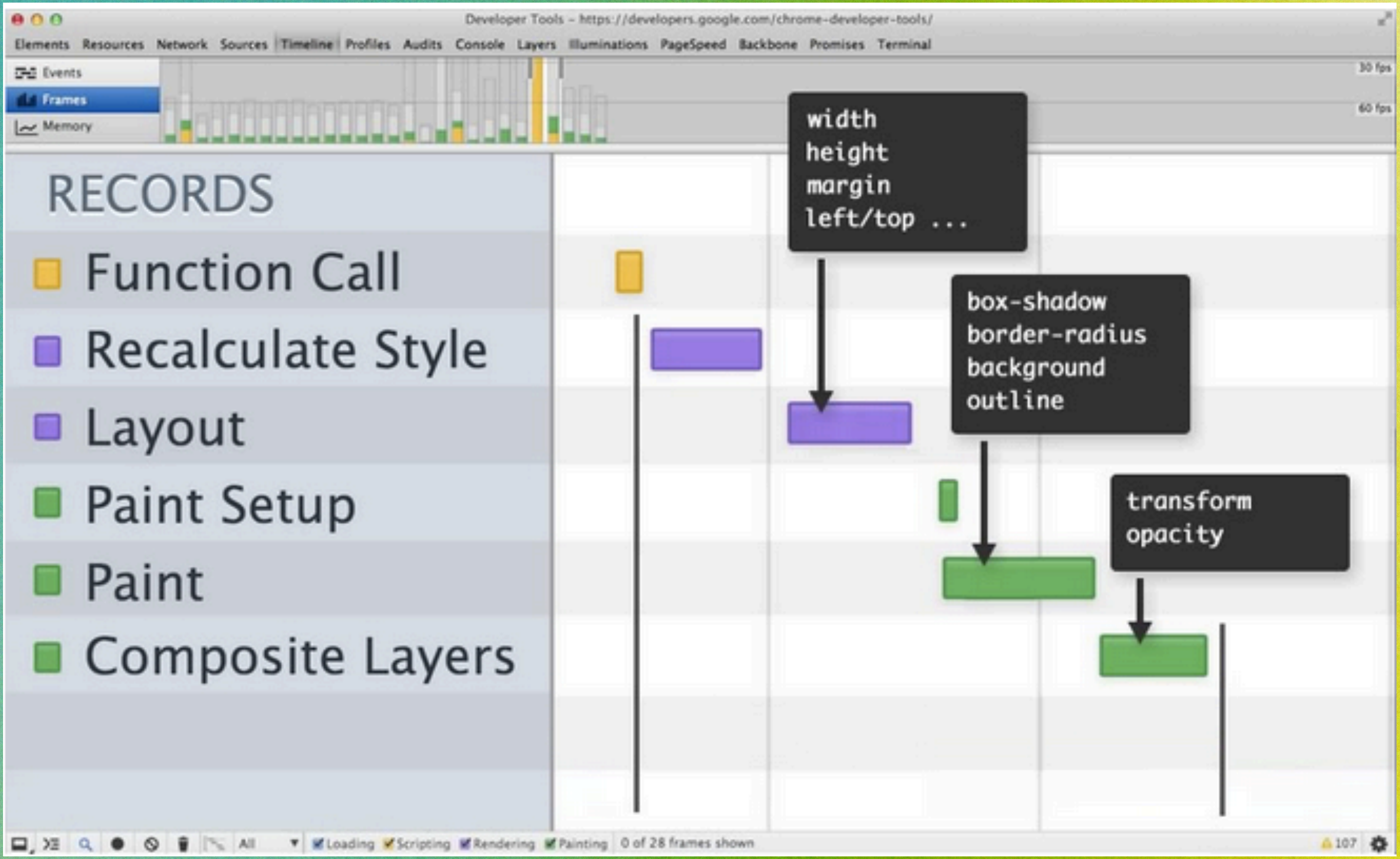
- ‘Jank free’
  - Avoid dropped frames, ideally achieving 60 frames per second [1]
- In our experience, bottleneck is hardly ever pure Javascript
- Usually layout/paint (we’ll come on to what this in a minute…)

[1] http://www.smashingmagazine.com/2013/06/10/pinterest-paint-performance-case-study/

Chrome Timeline

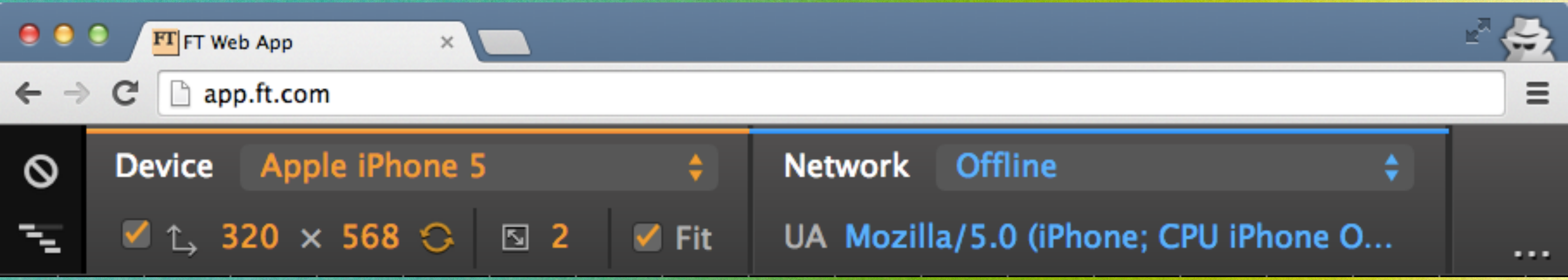


What Does This Show Us?


Developer Tools - https://developers.google.com/chrome-developer-tools/
Elements Resources Network Sources Timeline Profiles Audits Console Layers Illuminations PageSpeed Backbone Promises Terminal
Events
Frames
Memory
30 fps
60 fps
RECORDS
🟨 Function Call
🟪 Recalculate Style
🟪 Layout
🟩 Paint Setup
🟩 Paint
🟩 Composite Layers
width
height
margin
left/top ...
box-shadow
border-radius
background
outline
transform
opacity
All
Loading Scripting Rendering Painting
0 of 28 frames shown
107

From: http://www.html5rocks.com/en/tutorials/speed/high-performance-animations/

Raising Signal:Noise


FT Web App

app.ft.com

Device: Apple iPhone 5

320 × 568

2

✓ Fit

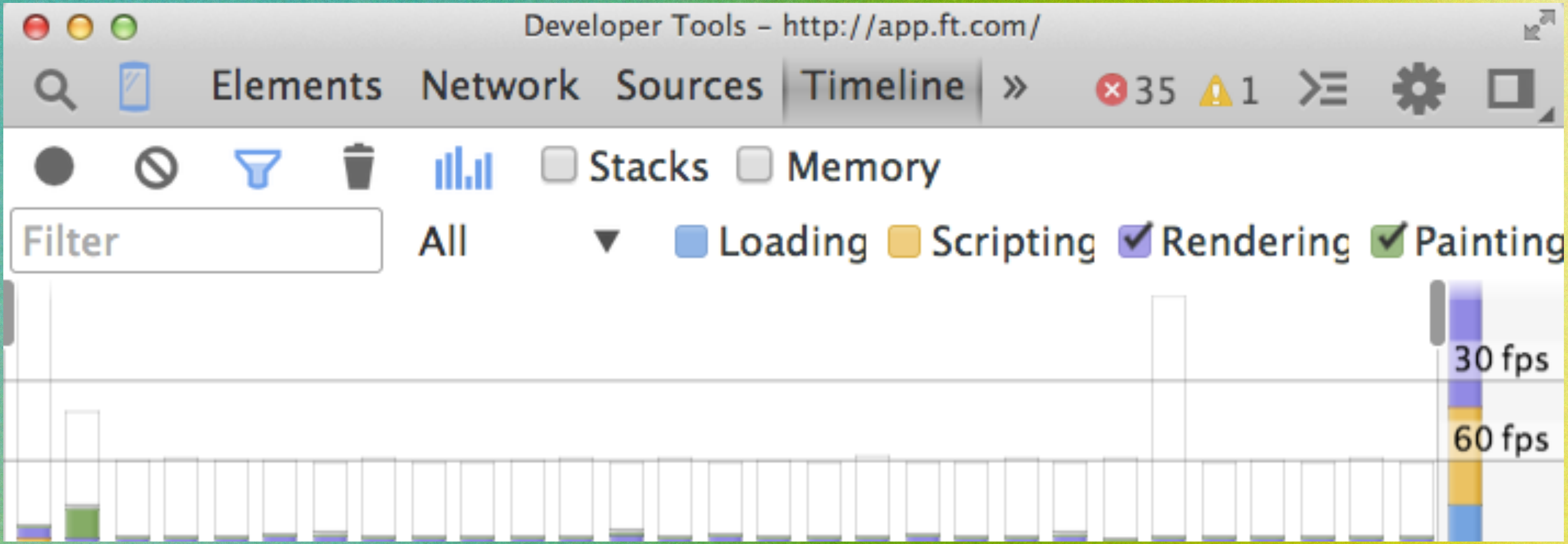
Network: Offline

UA: Mozilla/5.0 (iPhone; CPU iPhone O...

- Use Chrome in Incognito mode to protect from extensions
- Consider blocking network requests
- Beware mouse movement

Swiping on app.ft.com

github.com/ftlabs/ftscroller



Avoid Layout & Paint

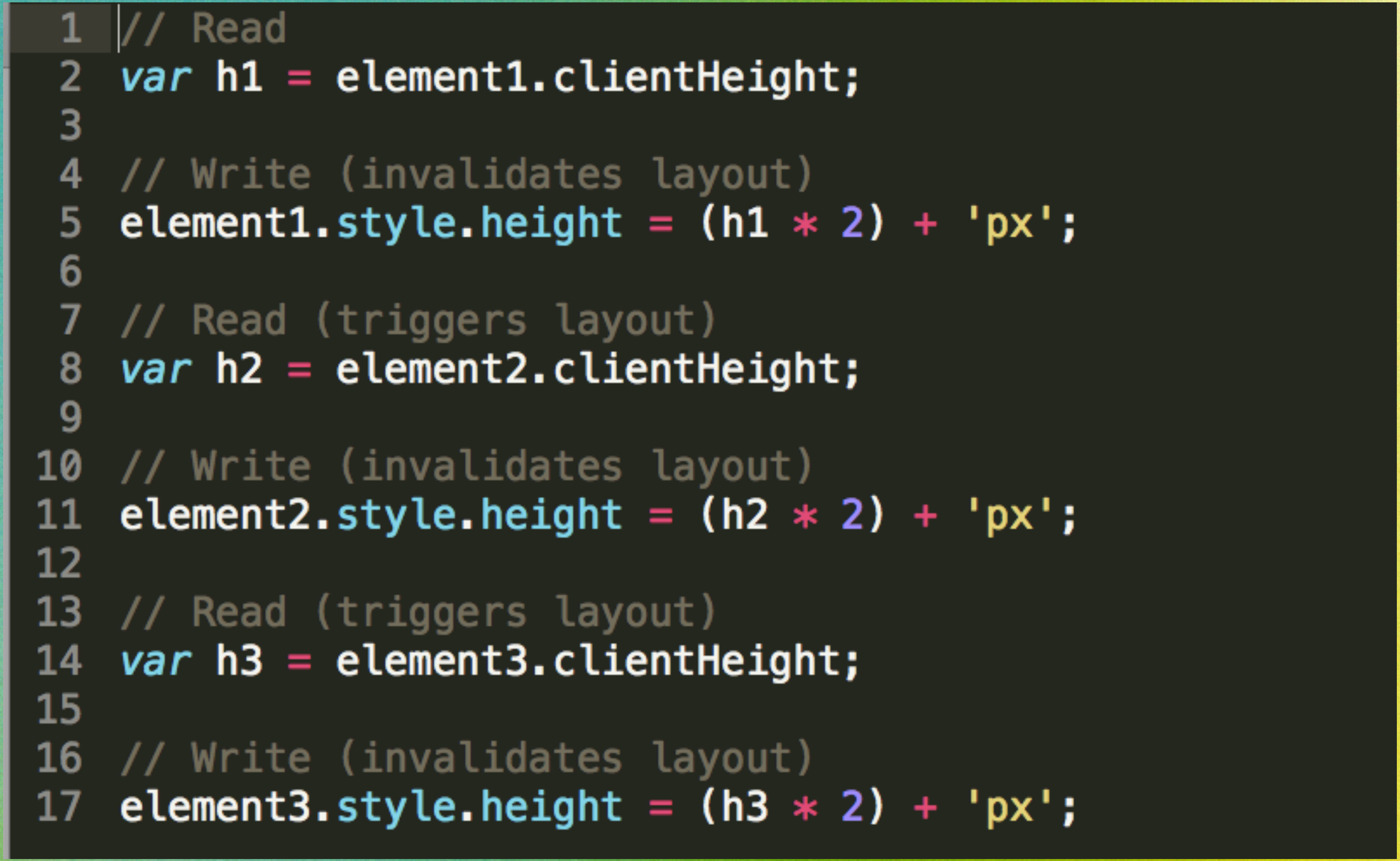
- Use hardware acceleration (GPU)
  - `translateZ()/translate3d()` hack [1]
  - Transforms (position, scale, rotation) and opacity are cheap if element is on it's own layer
  - Not a silver bullet - layers take up memory
- Future is declarative: `will-change` [2][3]

[1] http://aerotwist.com/blog/on-translate3d-and-layer-creation-hacks/
[2] http://aerotwist.com/blog/bye-bye-layer-hacks/
[3] https://dev.opera.com/articles/css-will-change-property/

Minimising Relayout

- Requiring a geometric value from the DOM forces it to layout synchronously if anything has invalidated layout
- Sounds kind of confusing…. Let’s see some code….

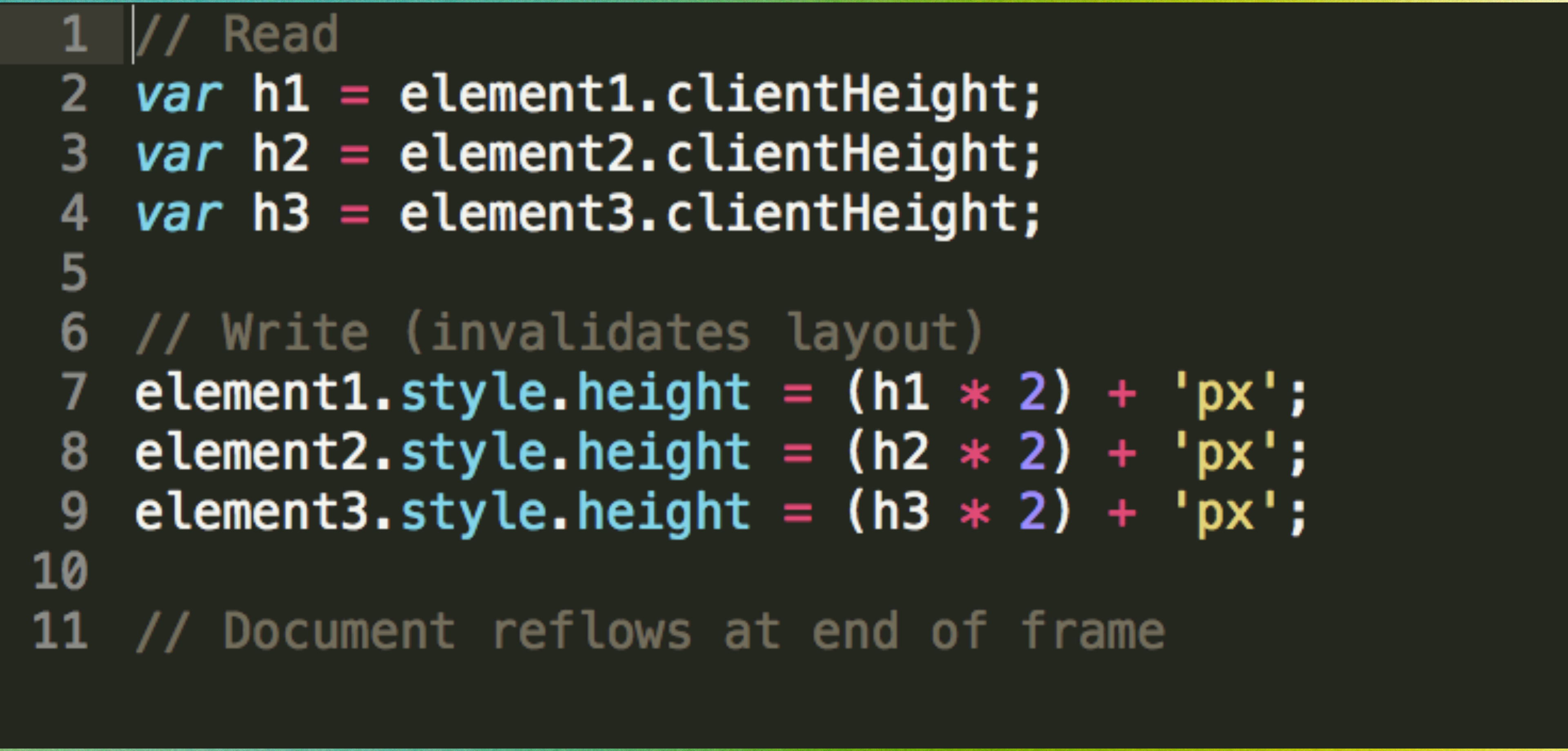
Layout ‘Thrashing’ [1]



```
// Read
var h1 = element1.clientHeight;

// Write (invalidates layout)
element1.style.height = (h1 * 2) + 'px';

// Read (triggers layout)
var h2 = element2.clientHeight;

// Write (invalidates layout)
element2.style.height = (h2 * 2) + 'px';

// Read (triggers layout)
var h3 = element3.clientHeight;

// Write (invalidates layout)
element3.style.height = (h3 * 2) + 'px';
```

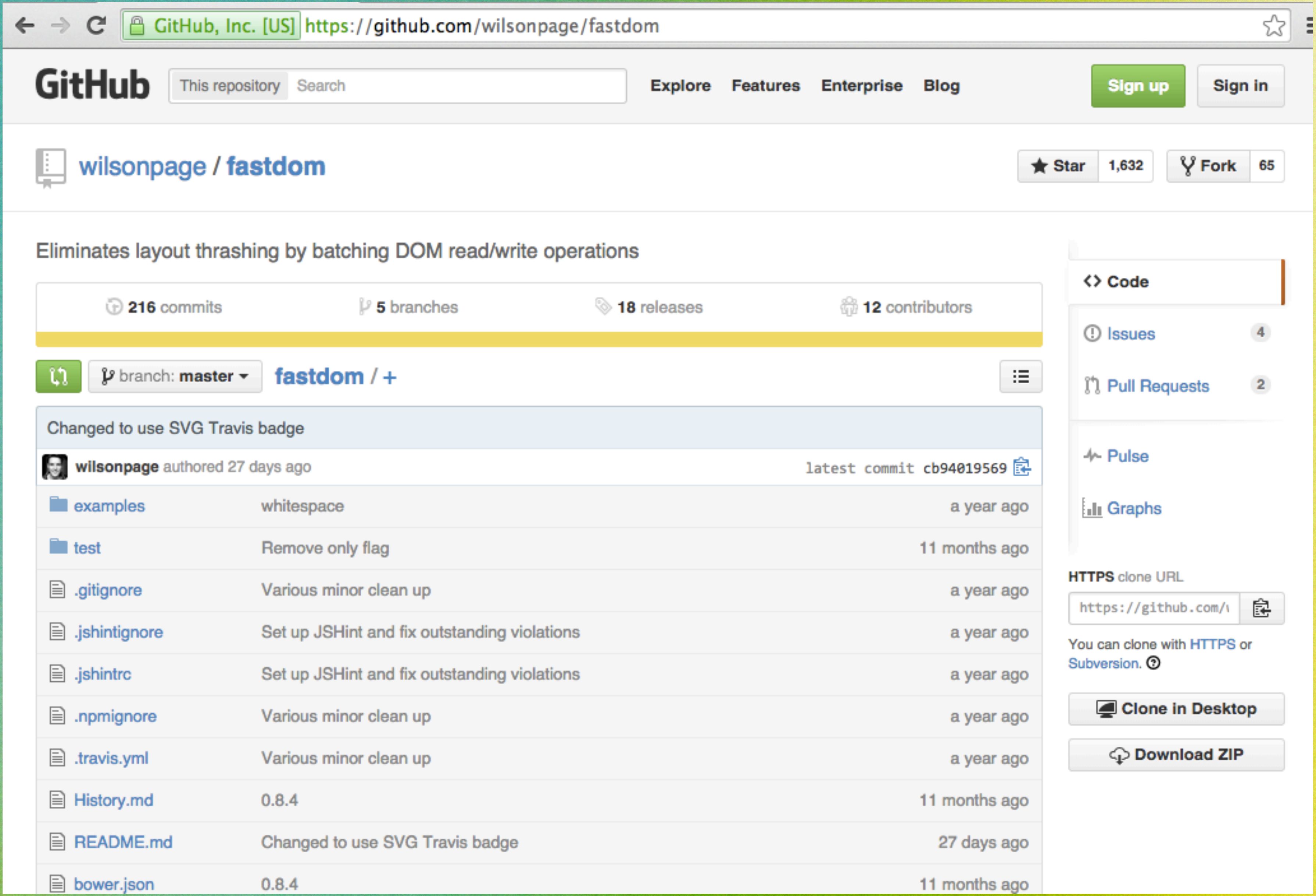
[1] http://wilsonpage.co.uk/preventing-layout-thrashing/

What We Ideally Want...



```
// Read
var h1 = element1.clientHeight;
var h2 = element2.clientHeight;
var h3 = element3.clientHeight;

// Write (invalidates layout)
element1.style.height = (h1 * 2) + 'px';
element2.style.height = (h2 * 2) + 'px';
element3.style.height = (h3 * 2) + 'px';

// Document reflows at end of frame
```

github.com/wilsonpage/fastdom



Eliminates layout thrashing by batching DOM read/write operations

216 commits · 5 branches · 18 releases · 12 contributors

Changed to use SVG Travis badge

wilsonpage authored 27 days ago · latest commit cb94019569

examples	whitespace	a year ago
test	Remove only flag	11 months ago
.gitignore	Various minor clean up	a year ago
.jshintignore	Set up JSHint and fix outstanding violations	a year ago
.jshintrc	Set up JSHint and fix outstanding violations	a year ago
.npmignore	Various minor clean up	a year ago
.travis.yml	Various minor clean up	a year ago
History.md	0.8.4	11 months ago
README.md	Changed to use SVG Travis badge	27 days ago
bower.json	0.8.4	11 months ago

HTTPS clone URL

https://github.com/

You can clone with HTTPS or Subversion.

Disclaimer

- This stuff is constantly changing – make sure you test your individual use case
- Keep up to date:
  - html5rocks.com/en/features/performance
  - jankfree.org/
  - Paul Lewis: aerotwist.com/

Offline

The Toolbox

- Application Cache (AppCache)
- LocalStorage
- IndexedDB
- (soon) ServiceWorker

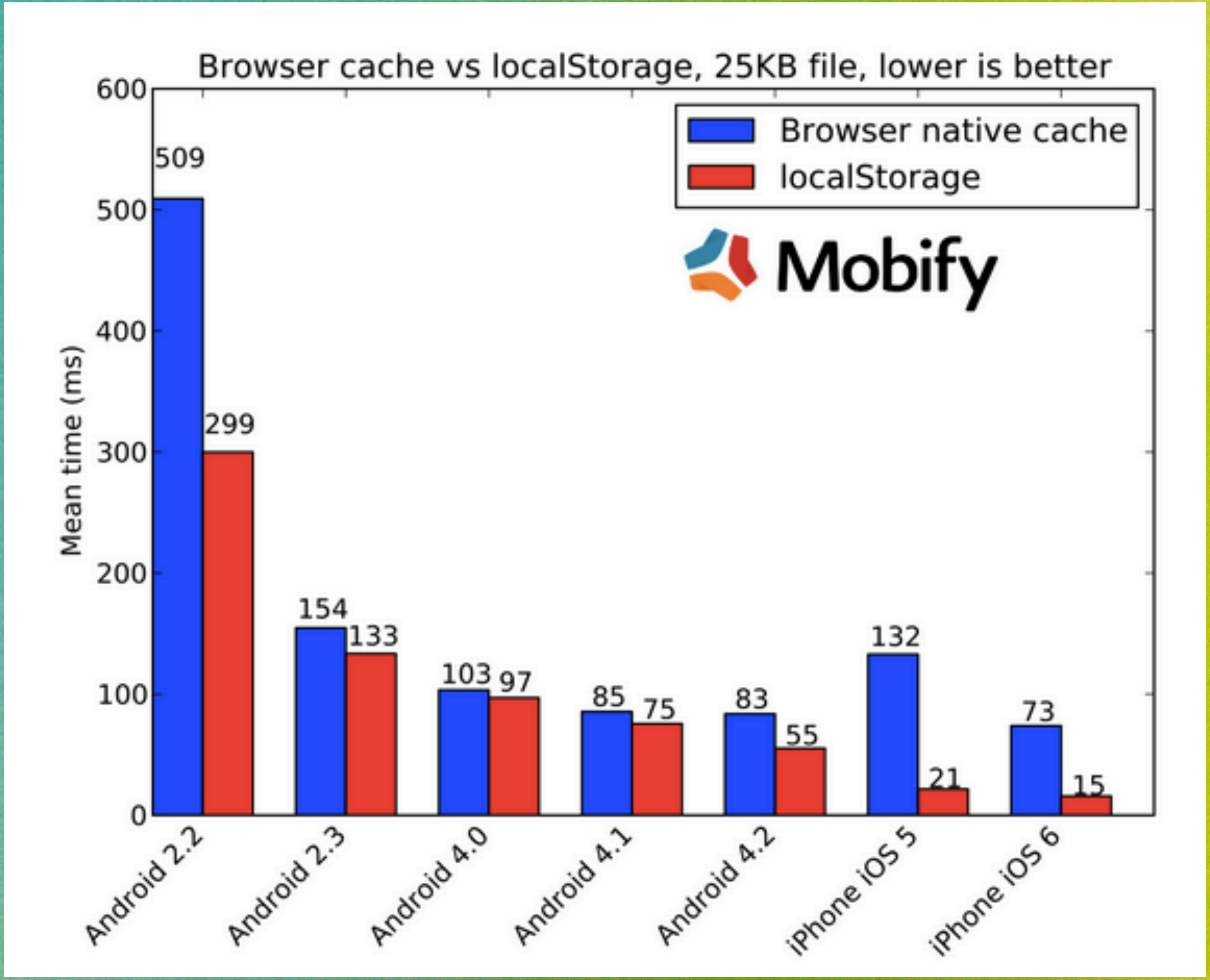
AppCache

- Well intentioned, but flawed:
  - Only updates if manifest file itself changed
  - Single change = complete redownload
  - Any failures = reversion to previous cached files
  - More: alistapart.com/article/application-cache-is-a-douchebag
- However, it *is* usable
  - We use it for bare minimum – bootstrap code, fonts, splash screen images

LocalStorage

- Simple API
- Fast

How Fast is localStorage?



http://www.mobify.com/blog/smartphone-localstorage-outperforms-browser-cache/

LocalStorage

- But:
  - Limited storage
  - Synchronous
  - File I/O for persistence means it can have variable performance
  - Odd behaviour in Safari private browsing
  - We use a lightweight async wrapper by Matt Andrews github.com/matthew-andrews/superstore

IndexedDB

- Indexable key value object store
  - We use this for articles and images
- Not supported everywhere - use polyfill [1] to support (long deprecated) WebSQL
- Managing versions and migrations can be awkward
- Documentation is variable

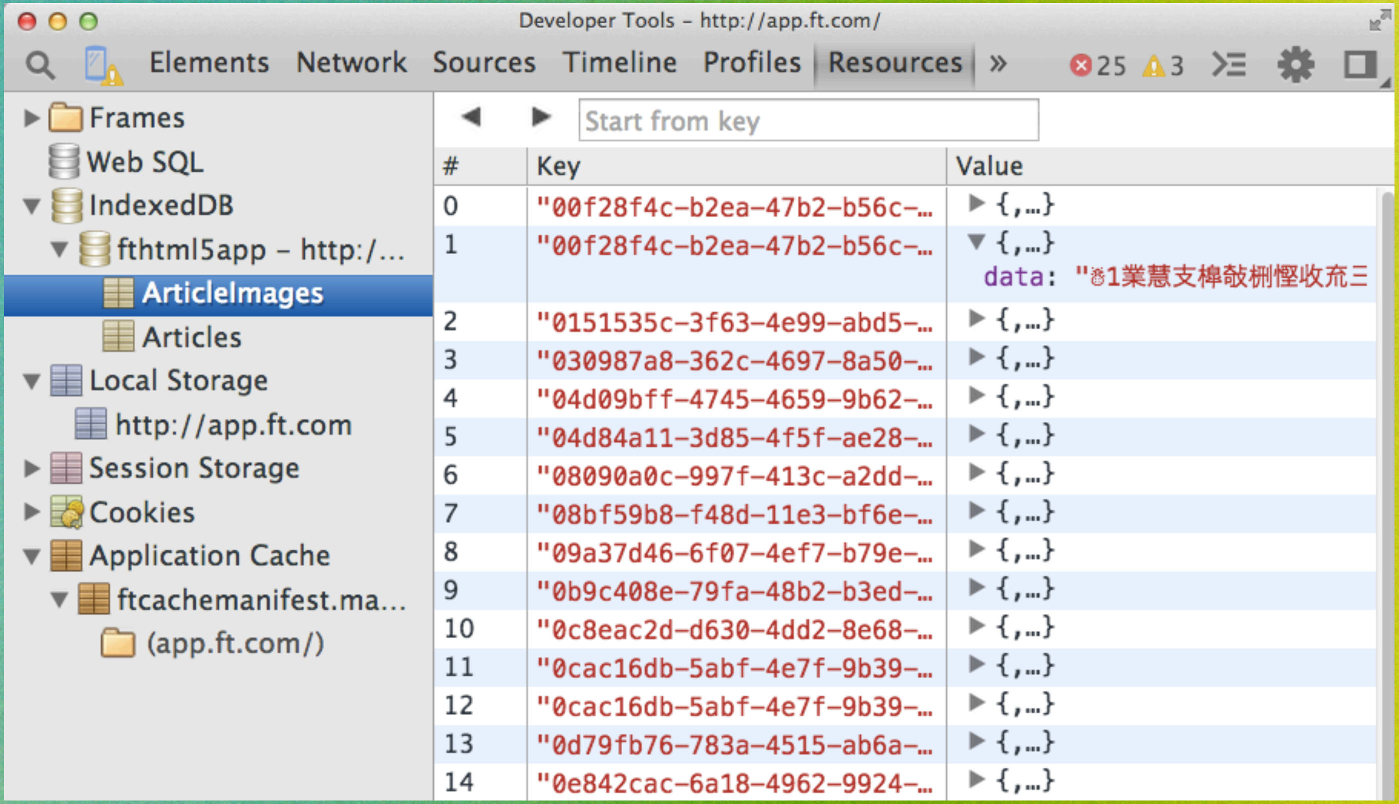
[1] http://nparashuram.com/IndexedDBShim/ or https://github.com/mozilla/localForage

How Much Can I Store?

- Depends on browser and permissions granted by user, see html5rocks.com/en/tutorials/offline/quota-research
- Getting more out of allowable storage by using UTF-8 instead of UTF-16: labs.ft.com/2012/06/text-re-encoding-for-optimising-storage-capacity-in-the-browser/
- Remember: need to base64 encode images to store them

Debugging / Inspecting

- DevTools



Developer Tools – http://app.ft.com/

```
▶ Frames
  Web SQL
▼ IndexedDB
  ▼ fthtml5app – http:/…
      ArticleImages
      Articles
▼ Local Storage
    http://app.ft.com
▶ Session Storage
▶ Cookies
▼ Application Cache
  ▼ ftcachemanifest.ma…
      (app.ft.com/)
```

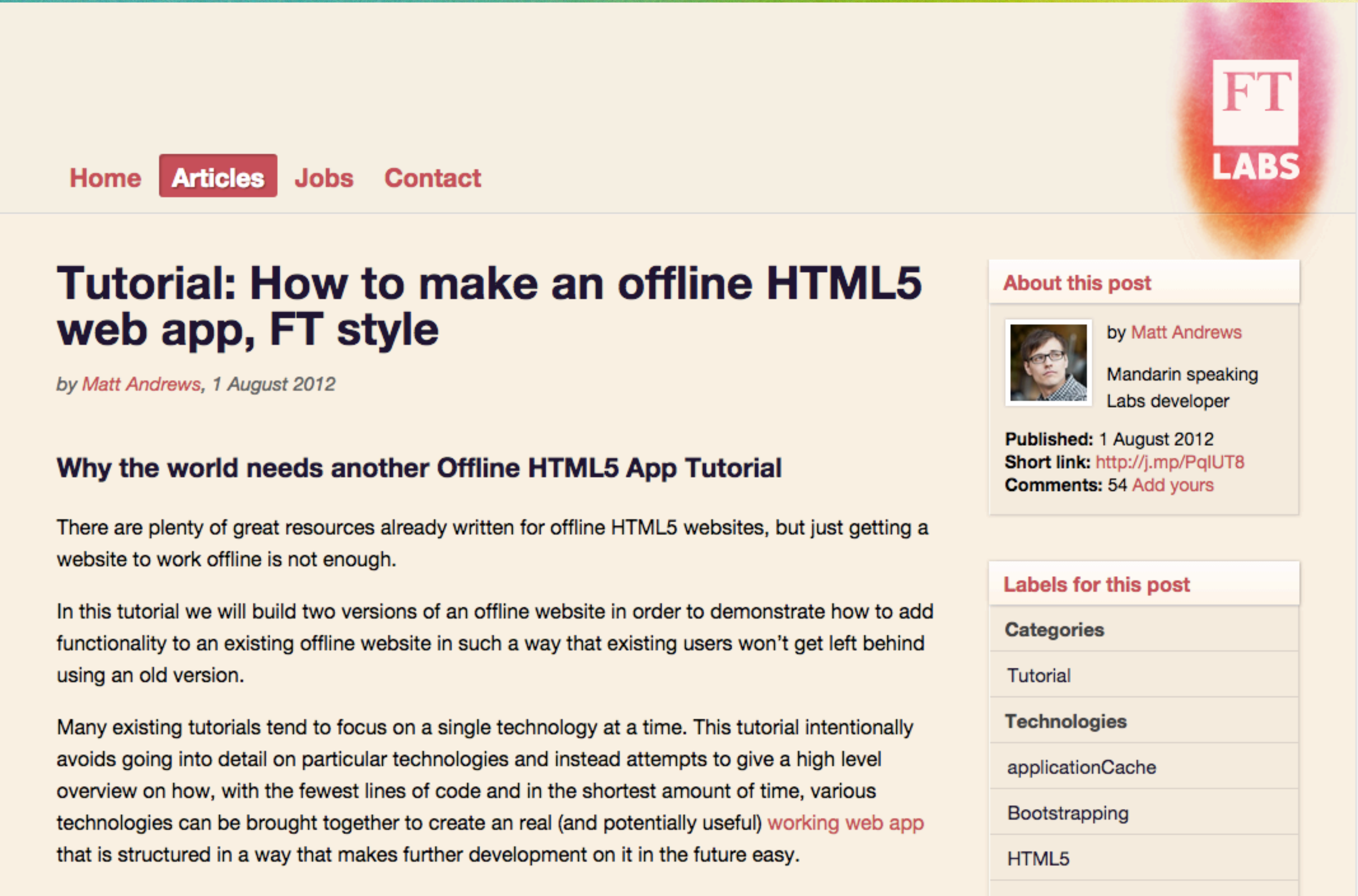
Start from key

#	Key	Value
0	"00f28f4c-b2ea-47b2-b56c-…	▶ {,…}
1	"00f28f4c-b2ea-47b2-b56c-…	▼ {,…} data: "[illegible]1業慧支榫敧梸慳收充三
2	"0151535c-3f63-4e99-abd5-…	▶ {,…}
3	"030987a8-362c-4697-8a50-…	▶ {,…}
4	"04d09bff-4745-4659-9b62-…	▶ {,…}
5	"04d84a11-3d85-4f5f-ae28-…	▶ {,…}
6	"08090a0c-997f-413c-a2dd-…	▶ {,…}
7	"08bf59b8-f48d-11e3-bf6e-…	▶ {,…}
8	"09a37d46-6f07-4ef7-b79e-…	▶ {,…}
9	"0b9c408e-79fa-48b2-b3ed-…	▶ {,…}
10	"0c8eac2d-d630-4dd2-8e68-…	▶ {,…}
11	"0cac16db-5abf-4e7f-9b39-…	▶ {,…}
12	"0cac16db-5abf-4e7f-9b39-…	▶ {,…}
13	"0d79fb76-783a-4515-ab6a-…	▶ {,…}
14	"0e842cac-6a18-4962-9924-…	▶ {,…}

Debugging / Inspecting

- Going a bit deeper:
  - chrome://appcache-internals
  - (confusingly) chrome://settings/cookies

labs.ft.com/2012/08/basic-offline-html5-web-app/



Tutorial: How to make an offline HTML5 web app, FT style

by Matt Andrews, 1 August 2012

Why the world needs another Offline HTML5 App Tutorial

There are plenty of great resources already written for offline HTML5 websites, but just getting a website to work offline is not enough.

In this tutorial we will build two versions of an offline website in order to demonstrate how to add functionality to an existing offline website in such a way that existing users won't get left behind using an old version.

Many existing tutorials tend to focus on a single technology at a time. This tutorial intentionally avoids going into detail on particular technologies and instead attempts to give a high level overview on how, with the fewest lines of code and in the shortest amount of time, various technologies can be brought together to create an real (and potentially useful) working web app that is structured in a way that makes further development on it in the future easy.

About this post

by Matt Andrews

Mandarin speaking Labs developer

Published: 1 August 2012
Short link: http://j.mp/PqlUT8
Comments: 54 Add yours

Labels for this post

Categories

Tutorial

Technologies

applicationCache

Bootstrapping

HTML5

ServiceWorker

- Replacement for AppCache, sits in the middle of browser and network
- Lots of good things:
  - Low level, granular control over browser network stack
  - Extensible
  - Introduces “Cache API” for storage
  - Can work with resources from any origin (opaque response)
  - Async
- But:
  - No access to LocalStorage
  - HTTPS only

Extending ServiceWorker

- The future?
  - Combine w/ background sync
  - Integrate with notifications and push API
- More:
  - How to use it today (published 24th Sept): jakearchibald.com/2014/using-serviceworker-today/
  - Google IO Talk: youtube.com/ watch?v=_yy0CDLnhMA
  - Browser support: jakearchibald.github.io/isserviceworkerready/

FT LABS

Thanks

Get in touch ☺ - @grahamhinchly / graham.hinchly@ft.com

Check out our open source - github.com/ftlabs

Come and work for us! - labs.ft.com/jobs