Making a website performant: Part I

Front end performance

Andy Callaghan
recollate.com


RECOLLATE

Overview

Part I

Front end performance

Common problems & fixes

Why bother?

Part II

Server side performance

Perceived performance

Planning to actually fix them

We want pages to start fast, and stay fast

What is performance?

“Load is not a single moment in time - it’s an experience that no one metric can fully capture.”

rc8.io/paint-timing

Key website measurements

Backend →	Frontend →	Perception
TTF Byte	TTF Paint	First Input Delay
Page weight	TTF Content	Framerate
Network speed	TTF Interactive	Animation

Some things I've found

'Load time' isn't useful

Measuring performance objectively is useful

Keeping focus on the user is hard

Getting your PM to give you time
to actually do the fixes is like really hard

Let's load a slow web page

4408

How American subtitle-writers caption a Yorkshirewoman saying “Huddersfield”. i.imgur.com

Submitted 13 hours ago by Dvdsmith2002

295 comments share save hide report

CasualUK

209,249 readers

Subscribe

Search

Submit Link Submit Text

This post was submitted on 04 Oct 2018

4,408 points (97% upvoted)

https://redd.it/9legmd

username password

remember me reset password LOGIN

Subreddit Info

2,456 currently online

We are looking for **lighthearted, fun** and **interesting** content and comments - whether it's an incredible picture you took of your hometown, a discussion you want to have on the best biscuit or a quirky UK focused article you want to share


>> I AM FROM HOODEZFIELD, IN
WEST YORKSHIRE.

Reddit Post

7.3

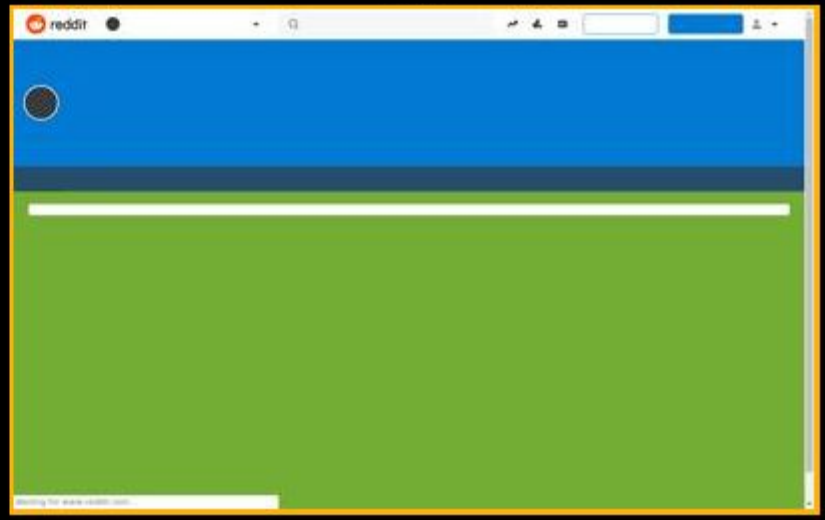
rc8.io/reddit-load

Frontend performance

TT First Paint

2.6s

2.7s

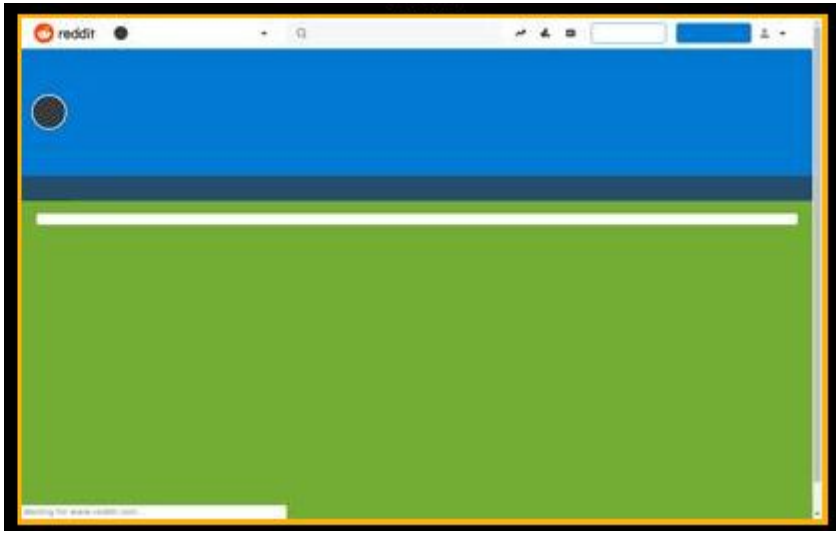


TTFP is 2.7s

First signs of life

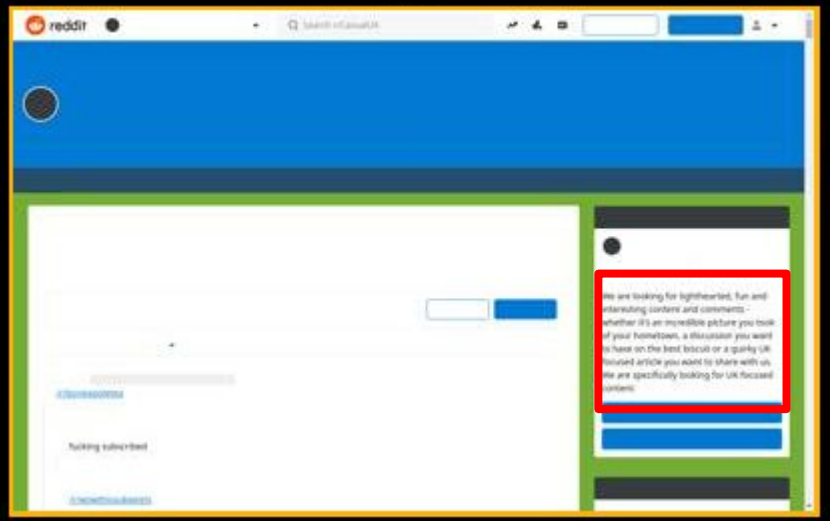
but...

No standard definition



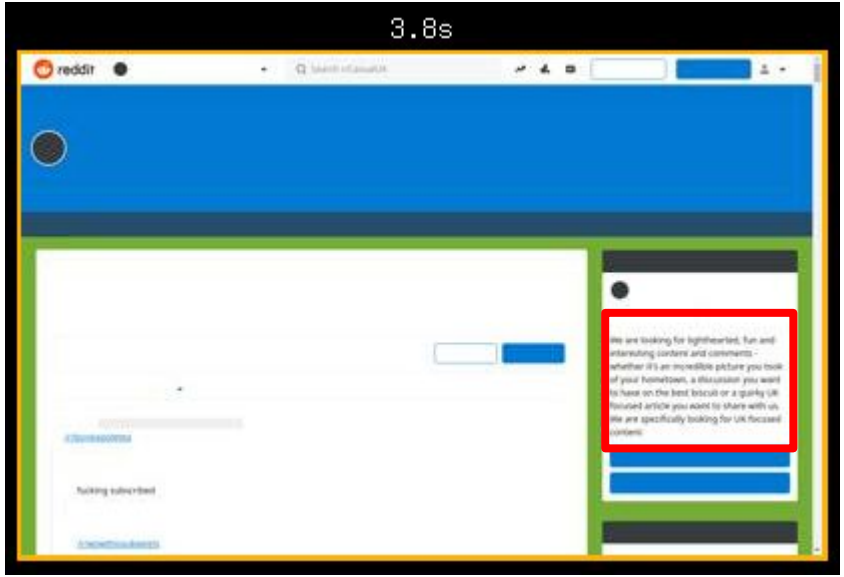
TT First Content

3.8s



TTFC is 3.8s

First ‘real content’
Different for every website
Easier to measure



First Paint/Render Common Issues

Page size, large DOM

Long critical rendering path

Complex CSSOM

Monolithic or render-blocking JS

TTFB (part II)

Universal CSS selectors

```
/* Selects all elements */

* {
  color: green;
}
```

Complicates CSSOM, longer to parse

rc8.io/universal-selector

First Paint/Content Common Fixes

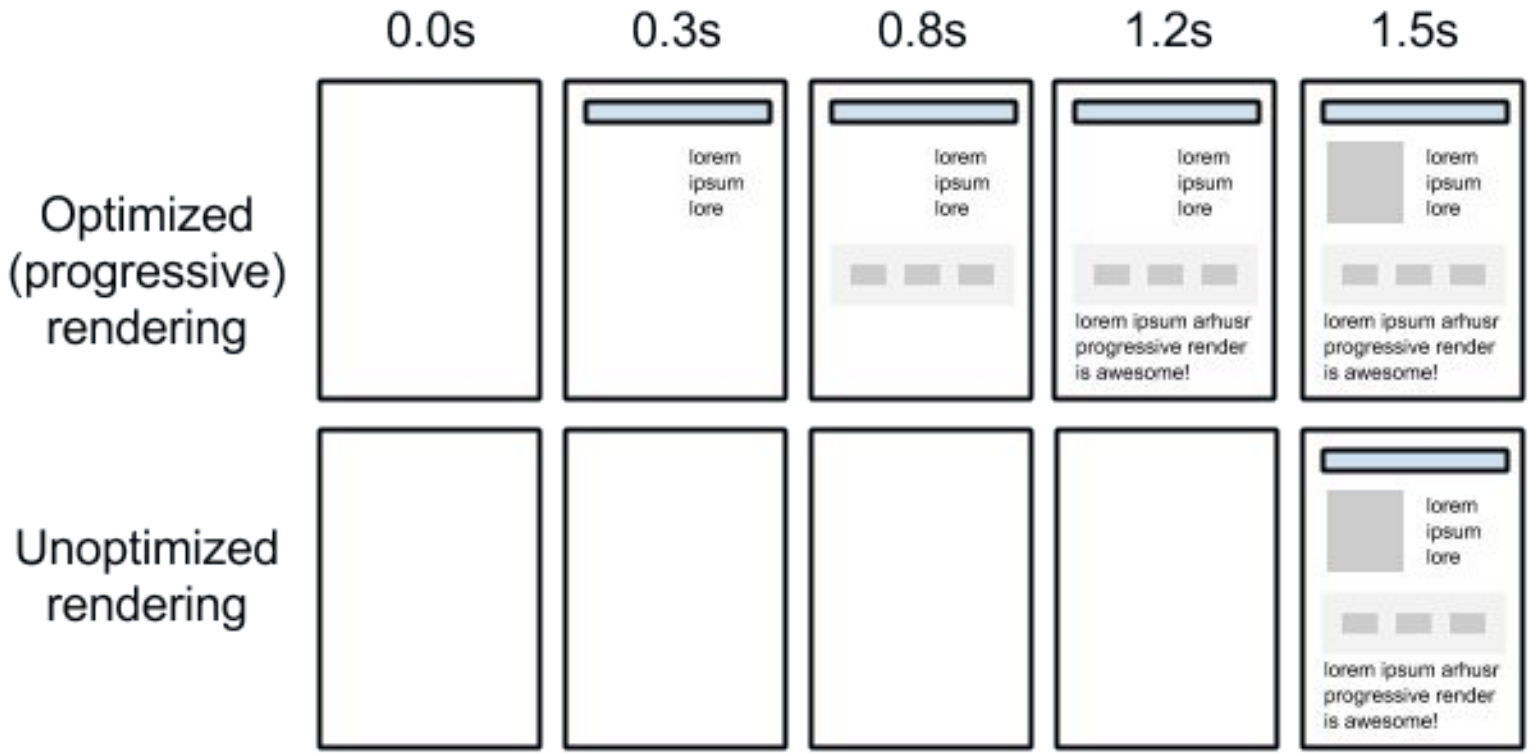
Finding your critical path, make it short

Reduce idle time in the thread

Progressive content rendering

Inline render blocking CSS, prioritise delivery

Async/defer non-critical JS



rc8.io/rendering-path

Webpack specific tuning

`mode 'production'`, uglify JS

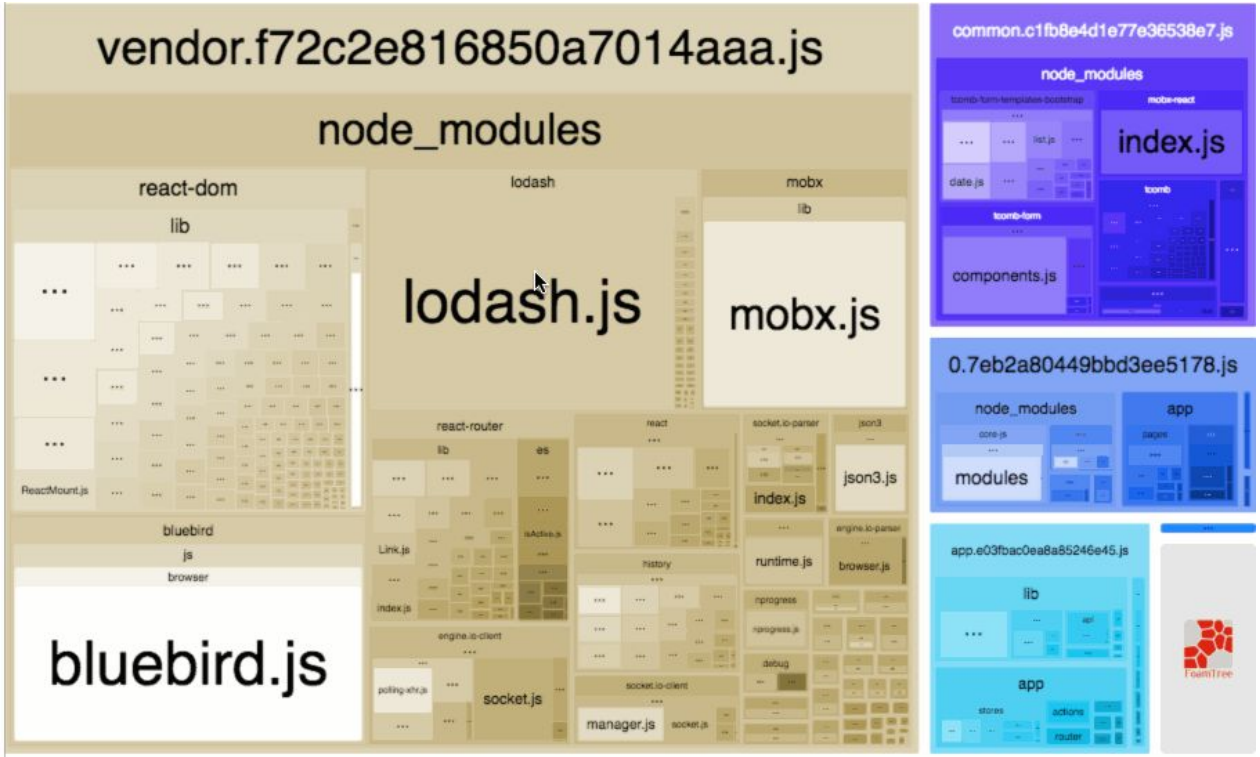
`Cache-Control: max-age=31536000`

Code splitting, only load the JS that's needed

Separate large sections - i.e. homepage, search

In ES6+ only import the functions you need

webpack-bundle-analyser



rc8.io/webpack-analyser

For example, `moment.js`

Loads all locales by default

400KB -> 100KB when they're removed

```
new webpack.ContextReplacementPlugin(/moment[\/\\]locale$/, /en/)
```

rc8.io/moment-js

...but be careful

"Styles at top, scripts at bottom" useless on large sites

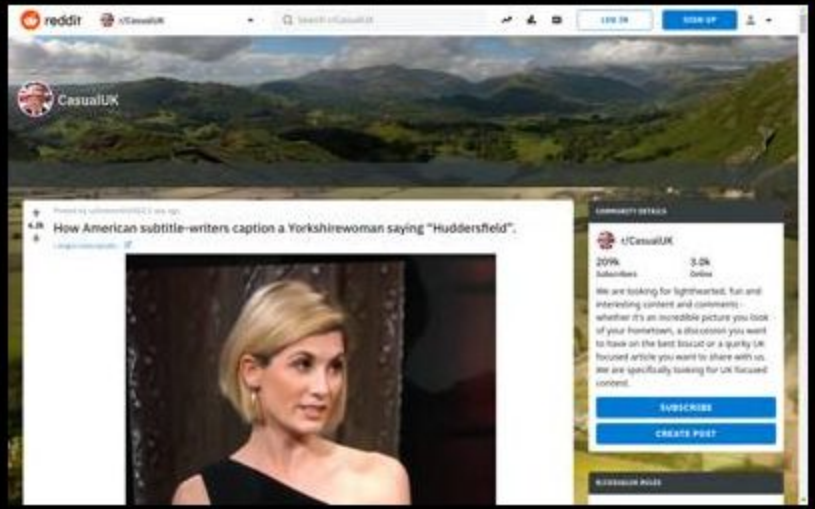
Don't reflow existing content once painted

Question why you need to lazy load

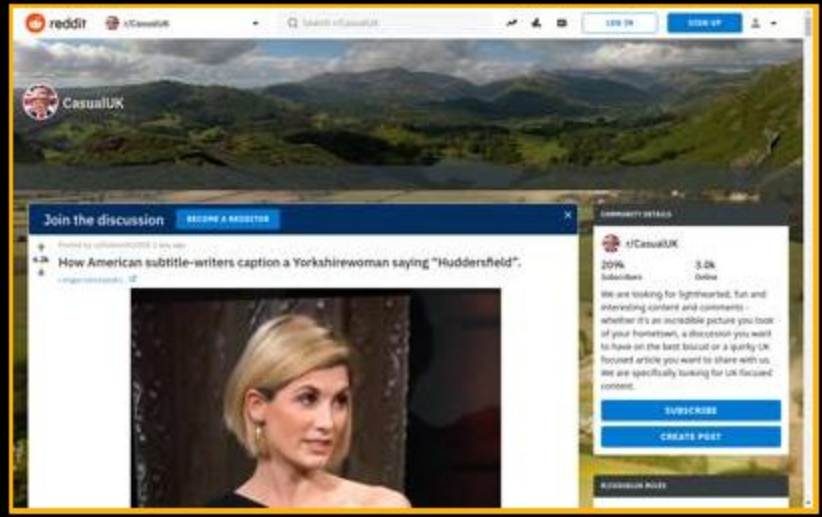
Above all: keep the user experience consistent

TT First Interactive

8.5s


reddit
r/CasualUK
LOG IN
SIGN UP
CasualUK
How American subtitle-writers caption a Yorkshirewoman saying "Huddersfield".
r/CasualUK
SUBSCRIBE
CREATE POST

8.6s


reddit
r/CasualUK
LOG IN
SIGN UP
CasualUK
Join the discussion
BECOME A REDDITOR
How American subtitle-writers caption a Yorkshirewoman saying "Huddersfield".
r/CasualUK
SUBSCRIBE
CREATE POST

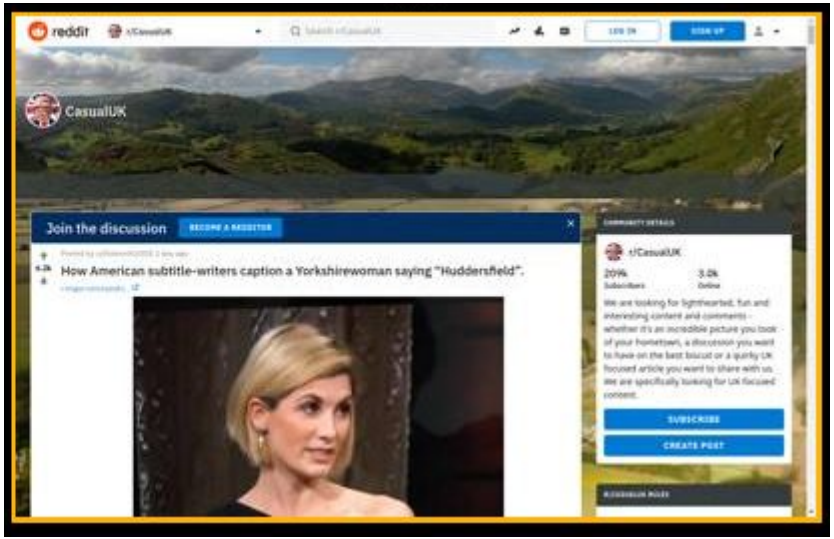
TTFI is 8.6s

Useful content

No 'jank' from now

Event handlers on most visible elements

rc8.io/ttfi-definition



Tracking TTFC in ES6

```
const observer = new PerformanceObserver((list) => {
  for (const entry of list.getEntries()) {
    console.log(entry.name) // => "first-paint"
  }
});
observer.observe({entryTypes: ['paint']}
```

First Interactive Common Issues

Third party JS

Web fonts

'Jankiness'

Long running JS

Large, unoptimised, non-visible images

CONTENT SIZES

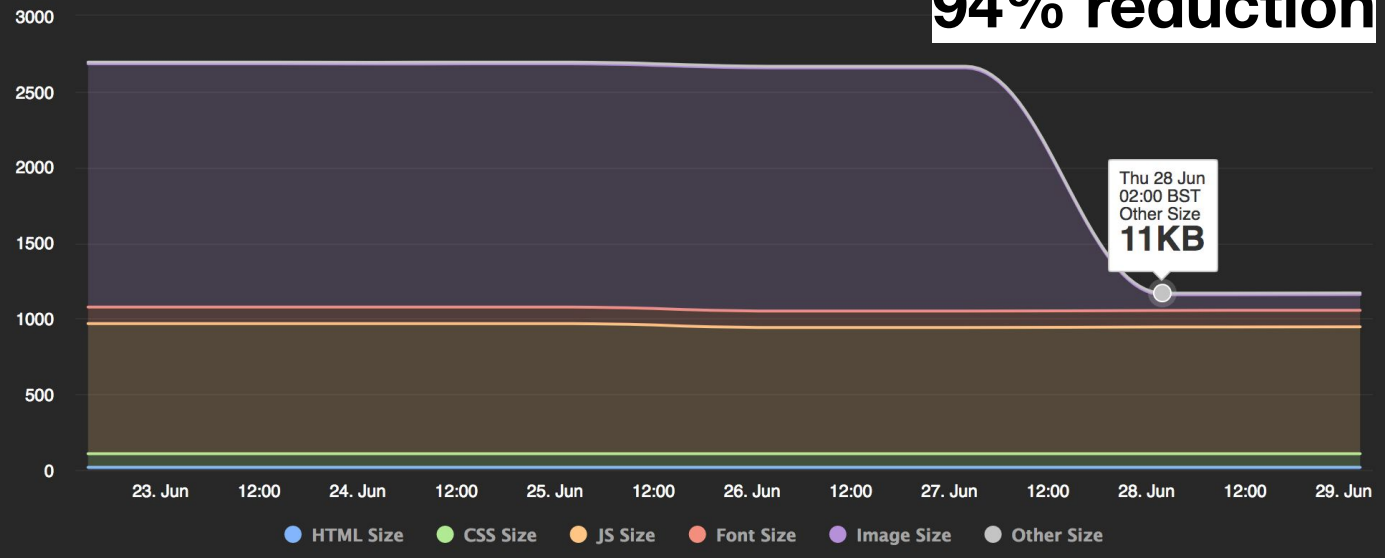
Notes

HTML Size	CSS Size	JS Size	Font Size	Image Size
17KB	89KB	839KB	109KB	1607KB

Other Size

11KB

Lazy loaded images
1600KB to 100KB
94% reduction



Javascript is the most expensive part of your site

Sam Saccone

@samccone

Follow

Time to first paint is meaningless when you take a poo on the main thread for 20 seconds.

5:26 AM - 16 Aug 2016

28 Retweets **92** Likes

11 28 92

rc8.io/poo-on-thread

Idle until urgent

Eager evaluation: slow start time, quick execution

Lazy evaluation: quick start time, slow execution

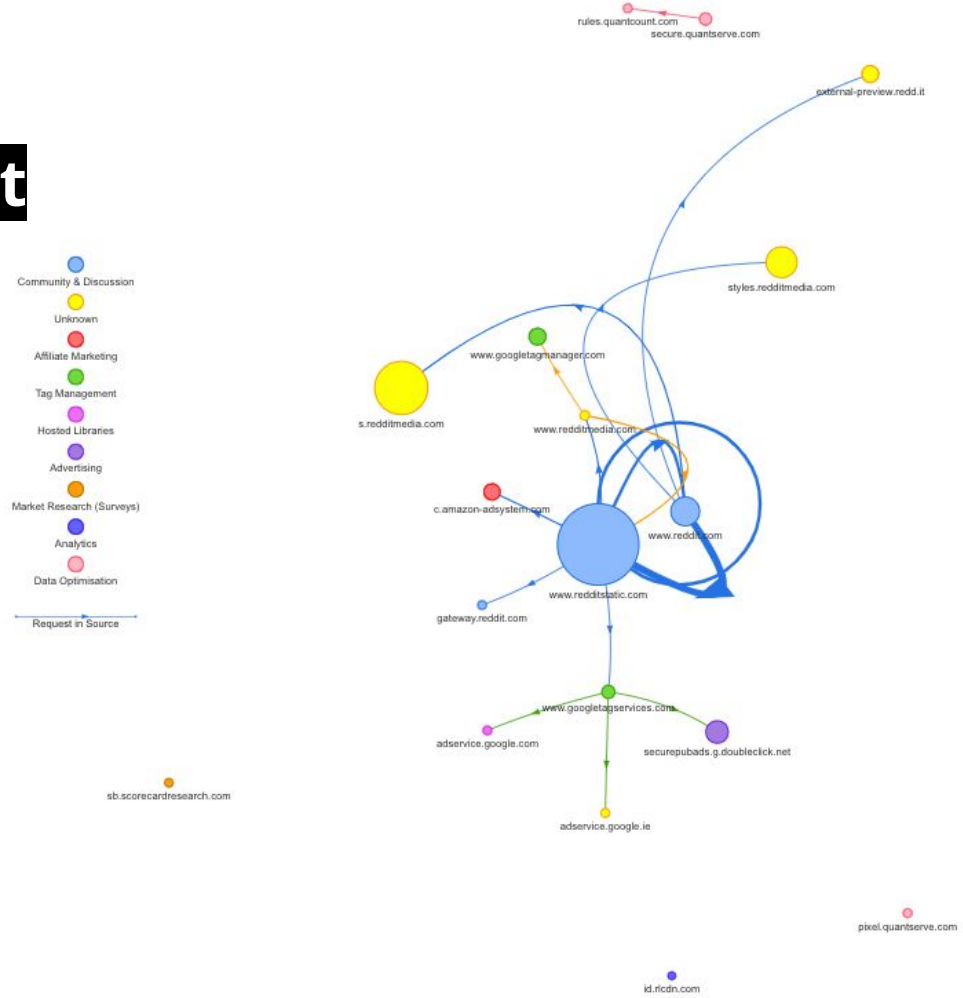
‘Idle until urgent’: Lazy, unless asked for before ready

Then, load it using a `requestIdleCallback`

rc8.io/state-of-js
rc8.io/request-idle-callback
rc8.io/idle-until-urgent

Third party JS can get out of hand

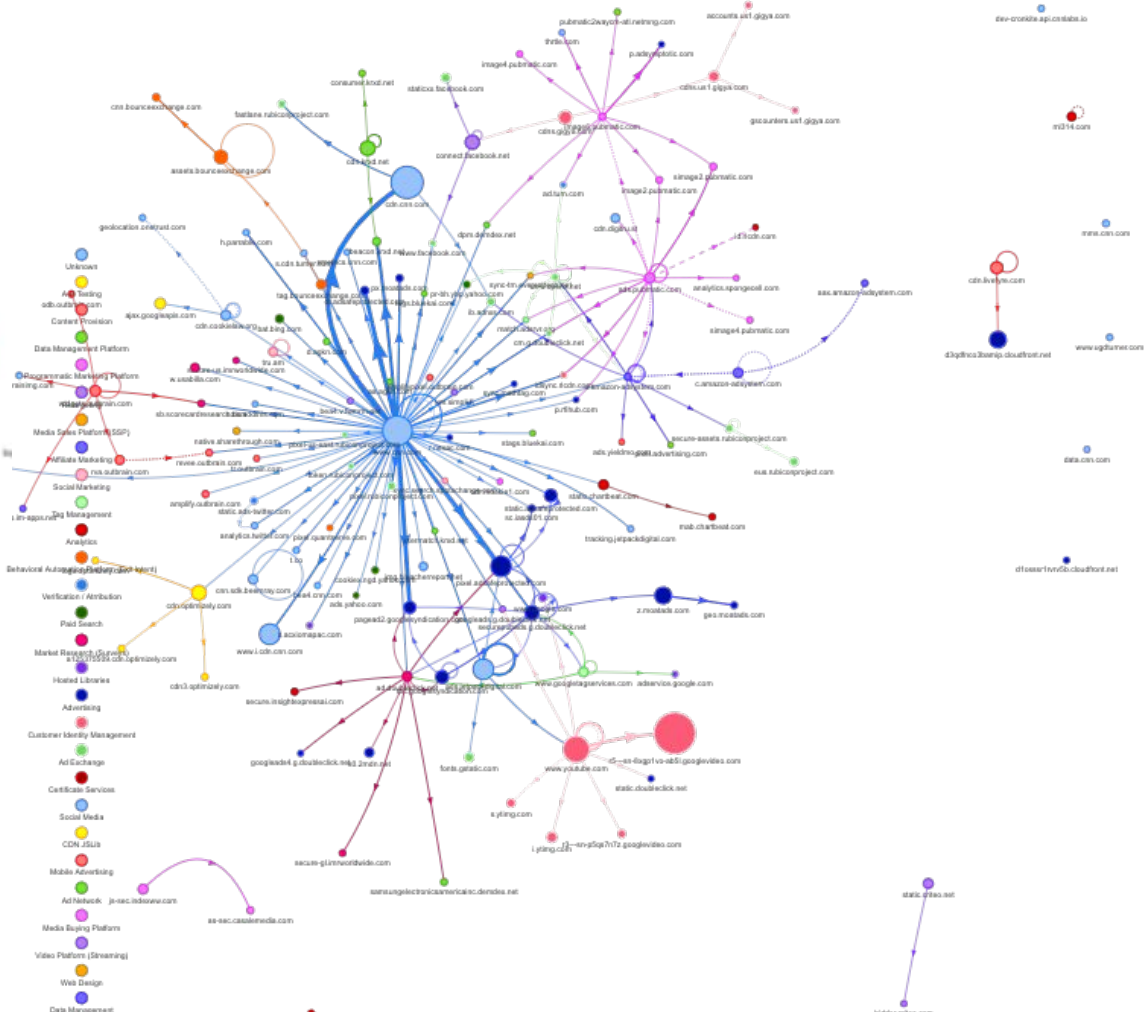
Reddit post

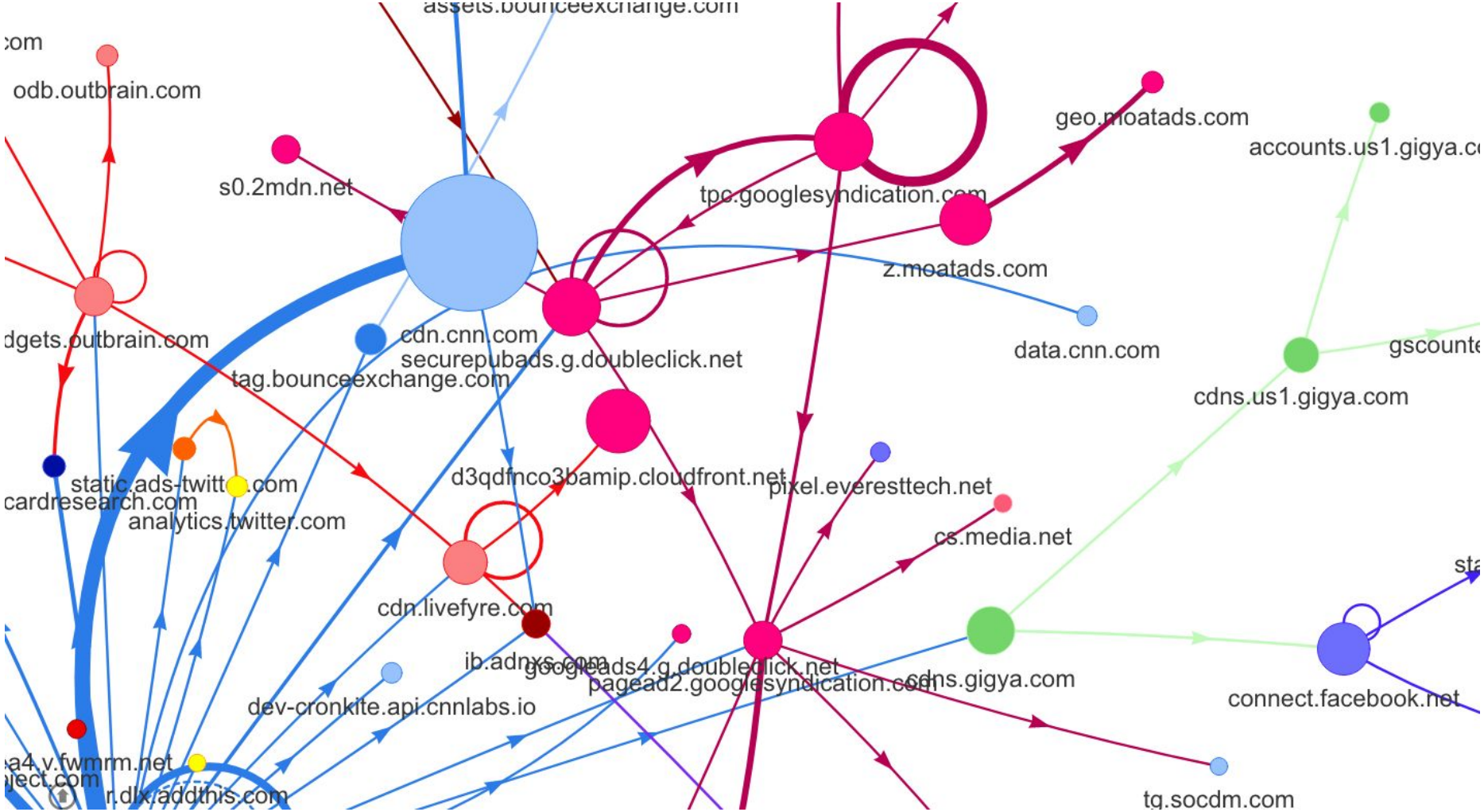


rc8.io/reddit-map

CNN.com


&$!#%


Unknown
Ad Testing
Content Provision
Data Management Platform
Programmatic Marketing Platform
Media Sales Platform (SSP)
Affiliate Marketing
Social Marketing
Tag Management
Analytics
Verification / Attribution
Paid Search
Hosted Libraries
Advertising
Customer Identity Management
Ad Exchange
Certificate Services
Social Media
CDN JSLib
Mobile Advertising
Ad Network
Media Buying Platform
Video Platform (Streaming)
Web Design


assets.bounceexchange.com
odb.outbrain.com
s0.2mdn.net
tpc.googlesyndication.com
geo.moatads.com
accounts.us1.gigya.c
z.moatads.com
dgets.outbrain.com
cdn.cnn.com
securepubads.g.doubleclick.net
data.cnn.com
tag.bounceexchange.com
cdns.us1.gigya.com
gscounte
static.ads-twitt
.com
cardresearch.com
analytics.twitter.com
d3qdfnco3bamip.cloudfront.net
pixel.everesttech.net
cs.media.net
cdn.livefyre.com
ib.adnxs.com
googleads4.g.doubleclick.net
pagead2.googlesyndication.com
cdns.gigya.com
connect.facebook.net
dev-cronkite.api.cnnlabs.io
a4.v.fwmrm.net
nect.com
r.dlx.addthis.com
tg.socdm.com

Google Tag Manager

Why bother?

Pinterest legacy site

TTFP 4.2s
TTFI 23s

Pinterest new PWA site

TTFP 1.8s
TTFI 5.6s

Time spent: +40%
Ad revenue: +44%
'Engagements': +60%

rc8.io/perf-stats

End of Part I

Making a website performant: Part II

Perceived performance

Andy Callaghan
recollate.com

Overview

Part I

Front end performance

Common problems & fixes

Why bother?

Part II

Server side performance

Perceived performance

Planning to actually fix them

Overview

Part I

Front end performance

Common problems & fixes

Why bother?

Part II

Server side performance

Perceived performance

Coffee and chat

Backend performance

TT First Byte

Queued at 0

Started at 1.89 ms

Chrome developer tools

Resource Scheduling	TIME
Queueing	1.89 ms
Connection Start	**TIME**
Stalled	3.31 ms
Request/Response	**TIME**
Request sent	0.41 ms
Waiting (TTFB)	468.55 ms
Content Download	354.95 ms
Explanation	**829.11 ms**

Queued at 0

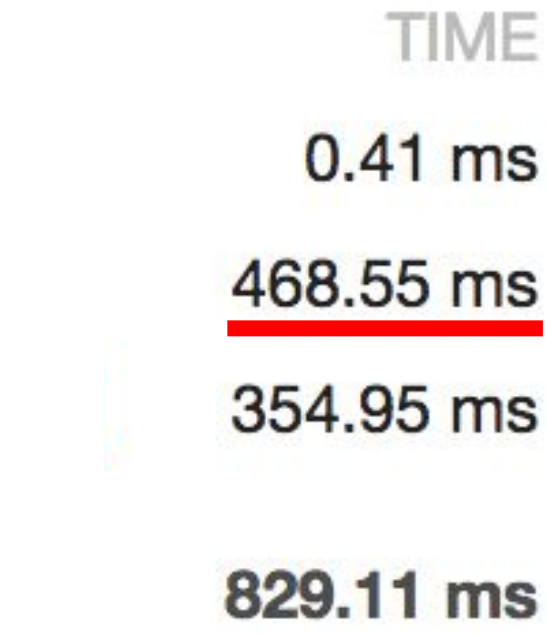
Started at 1.89 ms

Chrome developer tools

Resource Scheduling	TIME
Queueing	1.89 ms
Connection Start	**TIME**
Stalled	3.31 ms
Request/Response	**TIME**
Request sent	0.41 ms
Waiting (TTFB)	468.55 ms
Content Download	354.95 ms
Explanation	**829.11 ms**

TTFB is 468ms

Server responsiveness



Page weight & connection speed

Queued at 0

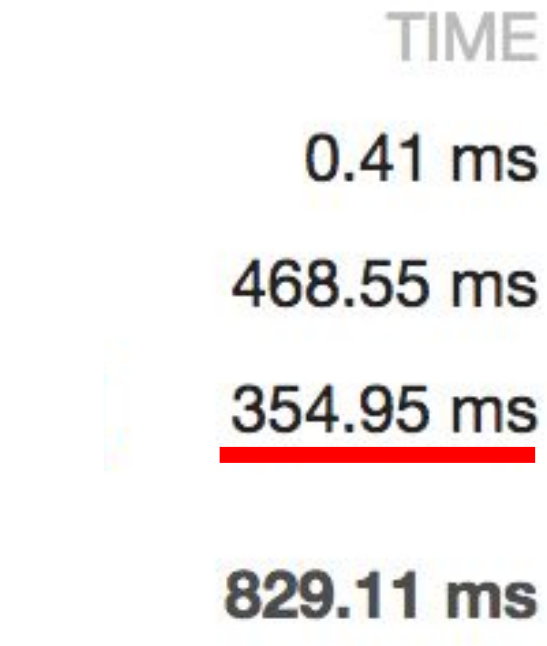
Started at 1.89 ms

Chrome developer tools

Resource Scheduling		TIME
Queueing		1.89 ms
Connection Start		**TIME**
Stalled		3.31 ms
Request/Response		**TIME**
Request sent		0.41 ms
Waiting (TTFB)		468.55 ms
Content Download		354.95 ms
Explanation		**829.11 ms**

HTML downloaded in 355ms

Varies across devices, networks, and pages



Fixing server performance issues

TTFB common issues

Physical distance between user & edge router

Server-side computation

Bad/restrictive infrastructure

Slow/shared hosting

Radio communication (e.g. crappy cellular 2/3G)

TTFB common fixes

Better and/or less dynamic code on a request

SQL EXPLAIN, good indexes, master/clones

More app servers, load balancing, app level caching

DNS preload, HTTP2, TLS 1.3, Service Workers

CDN delivery, ‘better’ hosting, dedicated CPU/memory

JAM stack

Templated views, compiled markup at build time

JS on the client-side for anything dynamic

APIs for integrations, authentication, search etc

Automated & atomic deploys, easy to rollback & scale

Fastest TTFB on the web

rc8.io/jam-stack

Service workers

A small script, running between client & server

Programmable proxy, control over network requests

Handles goodies like offline, push, caching strategy

No access to DOM, only `postMessage`

Using a service worker

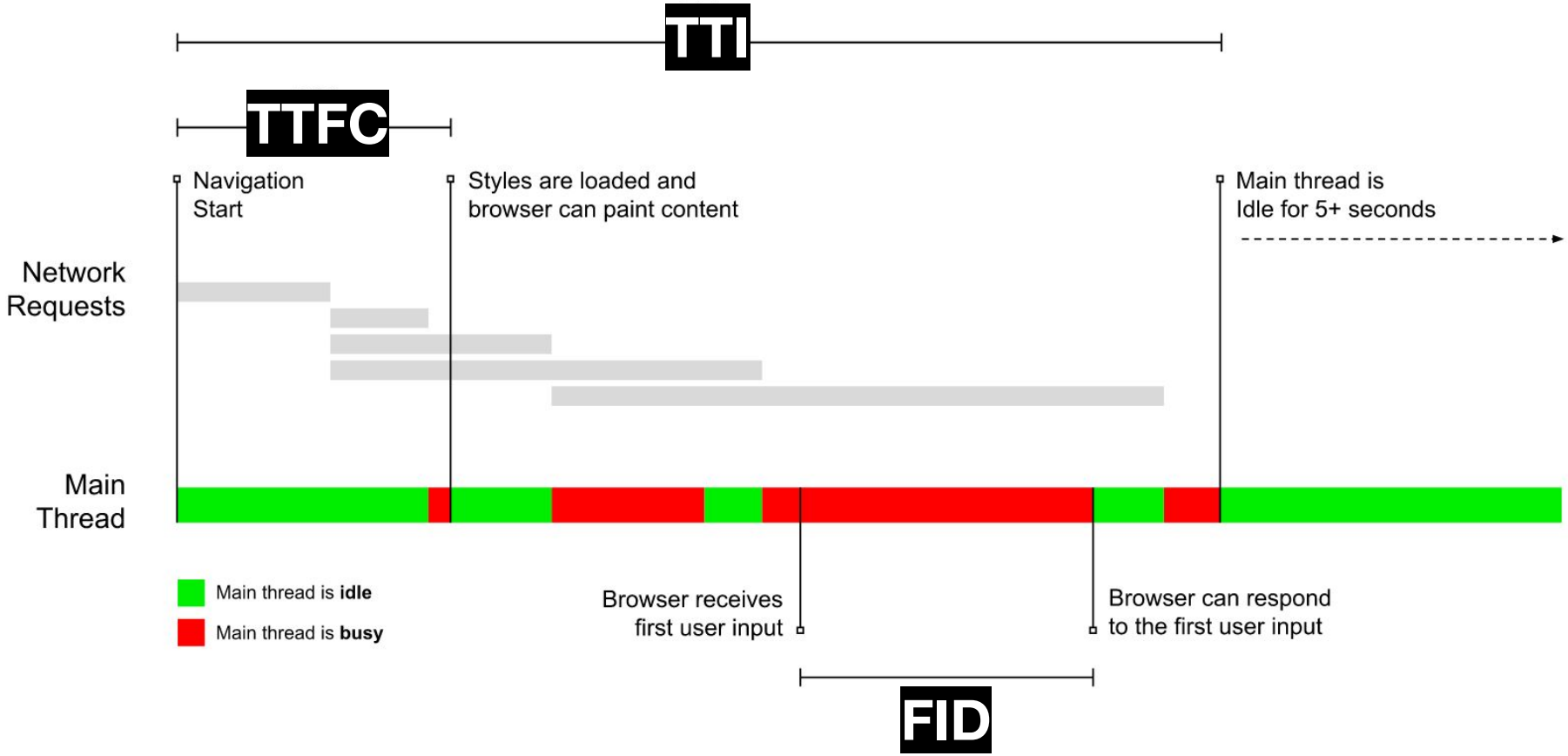
spotlight-pol2_hdpi.png	200	png	util.js:121	1.6 KB	161 ms
mapcnt6.png	200	png	util.js:124	2.4 KB	160 ms
Lato-Regular.woff2	200	font	celtic-manor-golf-hotel	(from ServiceWorker)	106 ms
Lato-Italic.woff2	200	font	celtic-manor-golf-hotel	(from ServiceWorker)	107 ms
Lato-Bold.woff2	200	font	celtic-manor-golf-hotel	(from ServiceWorker)	113 ms
GA collect?v=1&_v=j68&a=1364792363&t=e…	302	text/html	analytics.js:2	232 B	142 ms

Perceived performance

First Input Delay

“from when a user first interacts with your site, to the time when the browser is actually able to respond to that interaction.”

rc8.io/input-delay


TTI
TTFC
Navigation Start
Styles are loaded and browser can paint content
Main thread is Idle for 5+ seconds
Network Requests
Main Thread
Main thread is idle
Main thread is busy
Browser receives first user input
Browser can respond to the first user input
FID

First Input Delay

‘Real life’ user experience whilst loading

Needs client-side code to report

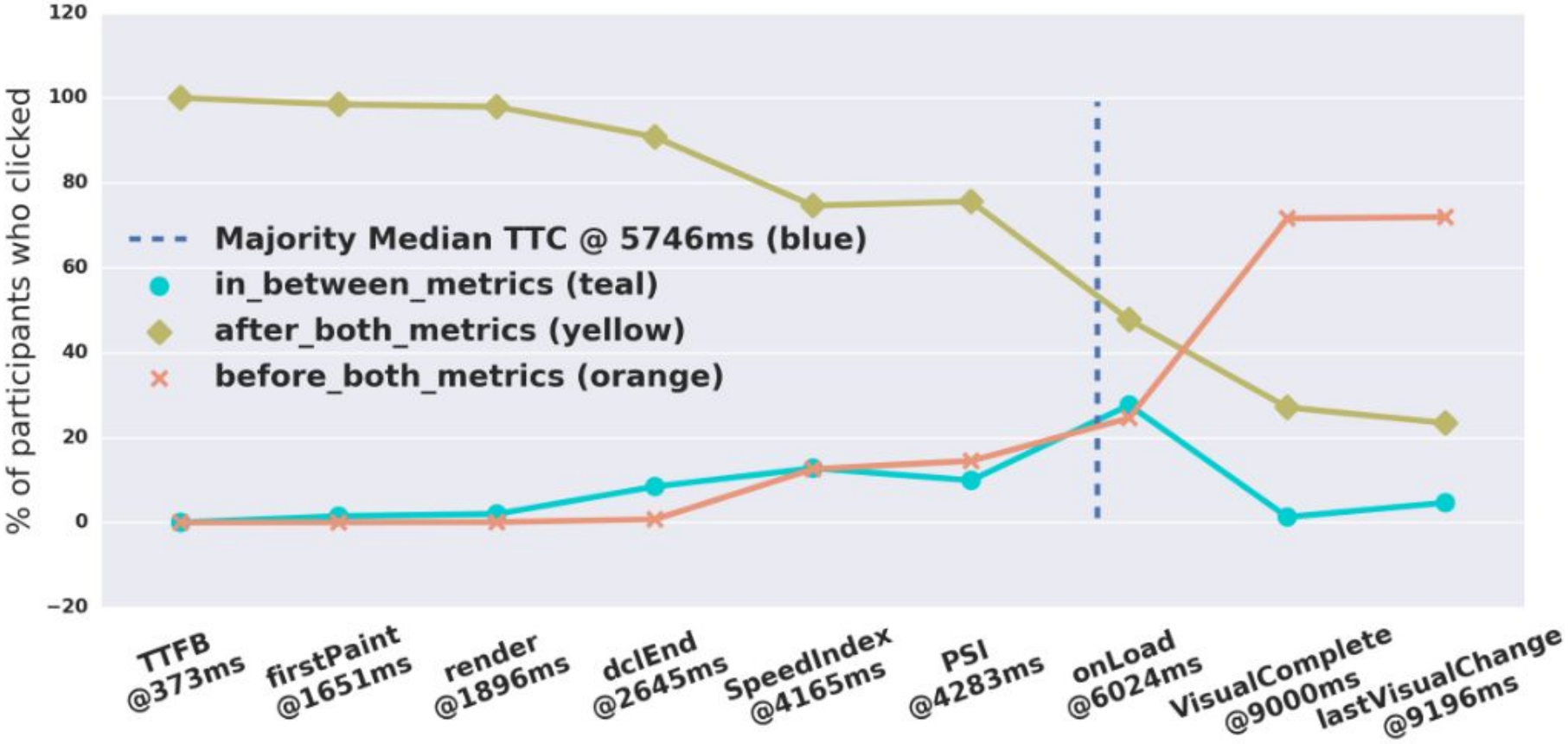
Scroll, click, 'main interaction' separately measured

rc8.io/median-fid

"metrics such as onLoad and TTFB fail to represent majority human perception"

rc8.io/perception-paper

User perception of "page is complete"



Actual performance

Expected performance

User experience

Expected performance

Context matters for perception

Web game

High FPS

10ms delay
is intolerable

Nausea

E-commerce

Changing framerate

100ms delay
is probably okay

$$$$

Perception matters

0 - 100ms	Instantaneous
100 - 300ms	Small, but perceptible
300ms - 1s	"*this machine is working*"
1 - 2s	Interruption to thoughtflow, context-switching
2+ seconds	Annoyance, rage clicking, loss of confidence
10+ seconds	"*I'll go and do something else*"

Ilya Grigorik, Google : rc8.io/delay-perception

Finish first paint within one second

Animation in CSS

CSS property changes can change the whole page

`opacity`, `transform` can be hardware accelerated

Avoid animating on other properties such as:
`border`, `padding`, `width`, `position`, `...`

Browsers try to guess compositing optimisations

max-height

max-width

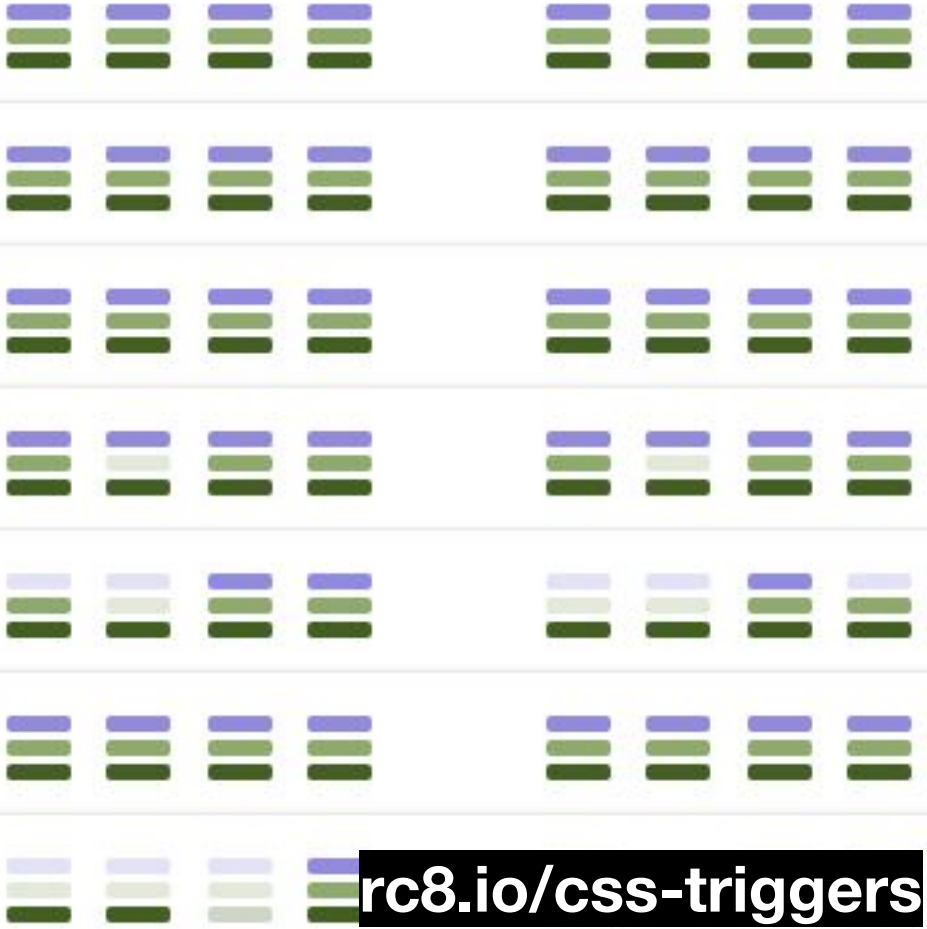
min-height

min-width

opacity

order

orphans



CSS will-change

Give browser hints, but use very sparingly (if at all)

```
.fading-thing {
  will-change: opacity;
  transform: opacity(1); // For IE/Edge
}

.fading-thing:hover{ opacity(0.5) }
```

User experience

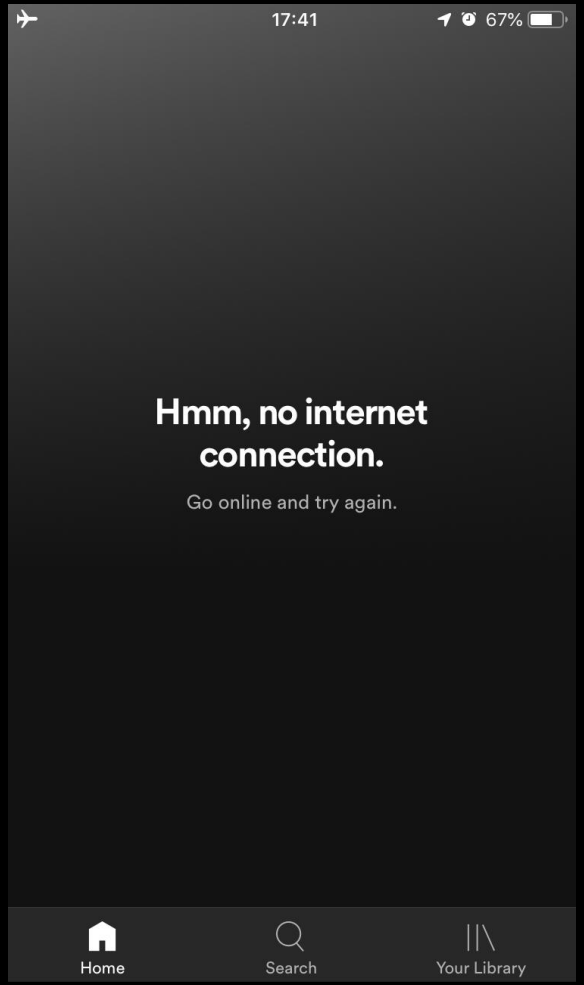
Mobile ≠ a device

‘Mobile’ describes your user

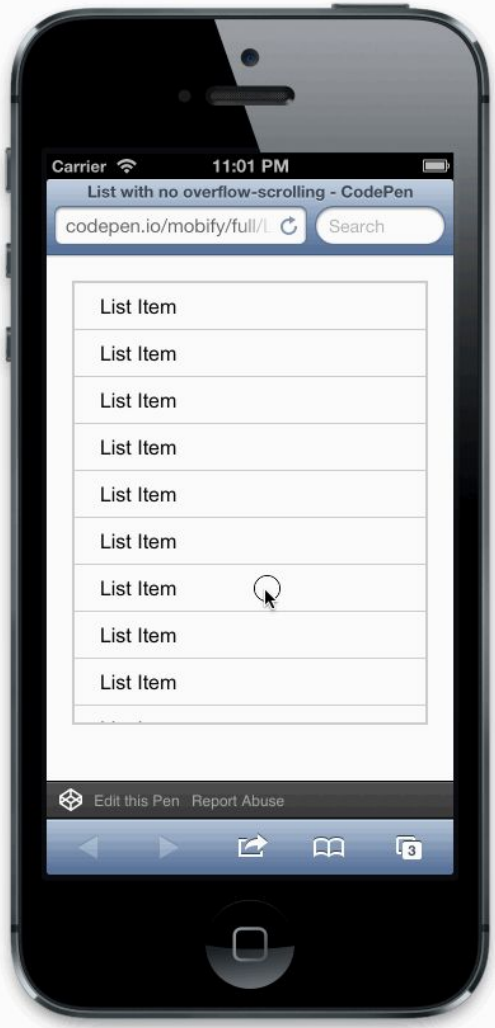
People that are ‘mobile’ have different expectations

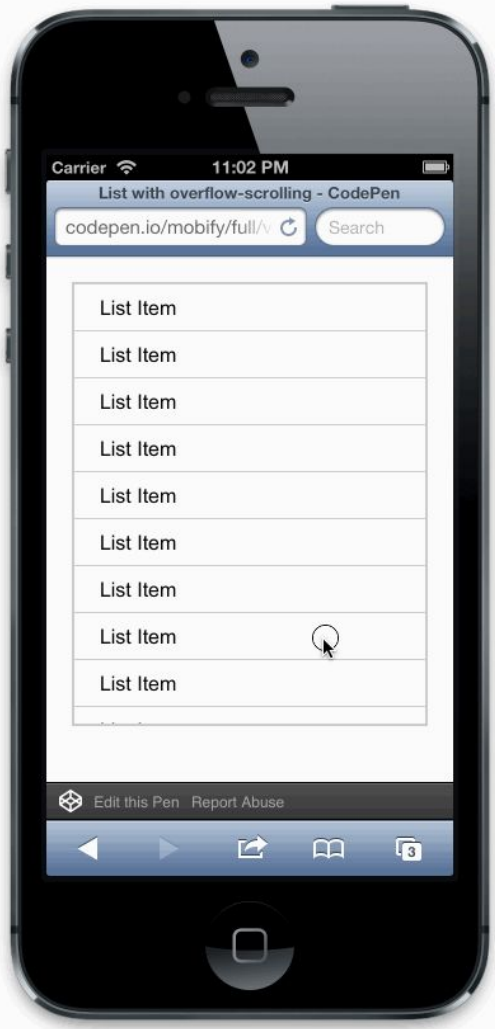
People that are walking, anxious, young or rushed perceive your site as slower

rc8.io/perception-paper



Spotify is offline

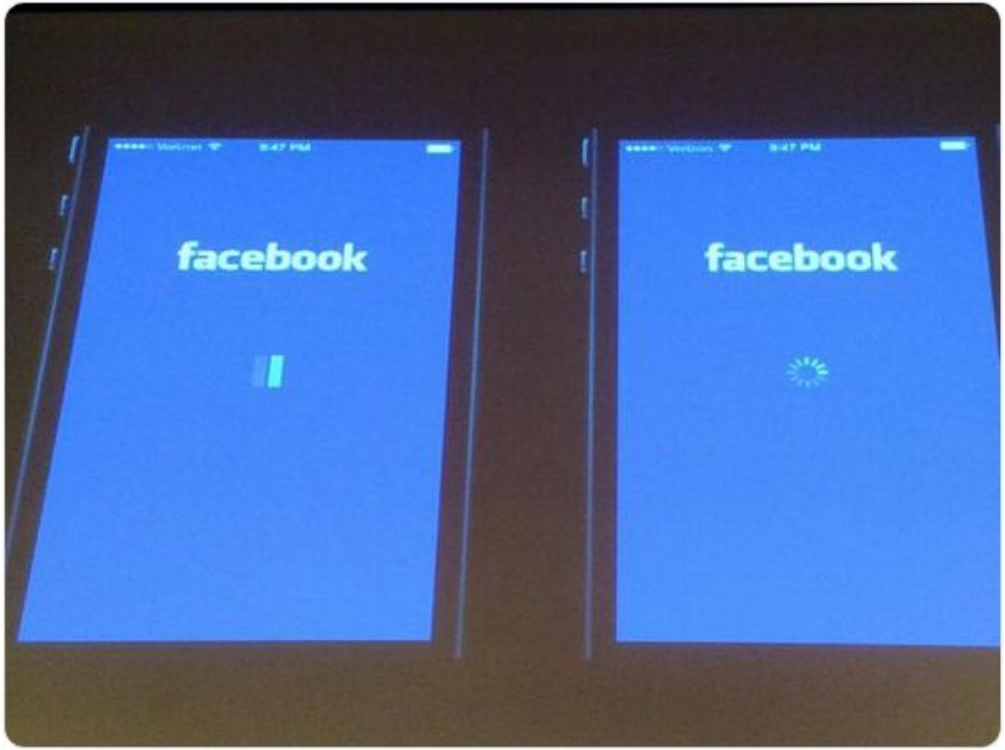

Carrier
11:01 PM
List with no overflow-scrolling - CodePen
codepen.io/mobify/full/l
Search
List Item
List Item
List Item
List Item
List Item
List Item
List Item
List Item
List Item
Edit this Pen
Report Abuse


Carrier
11:02 PM
List with overflow-scrolling - CodePen
codepen.io/mobify/full/v
Search
List Item
List Item
List Item
List Item
List Item
List Item
List Item
List Item
List Item
Edit this Pen
Report Abuse

deeje
@deeje

Following

#renio interesting! Facebook did A/B testing to determine that users blamed FB on left, iOS on right, for slowness.



Mobile users want a native UI

Inertia scrolling

Touch 'press' `document.addEventListener("touchstart", ...)`

Pre-emptive/optimistic UI changes

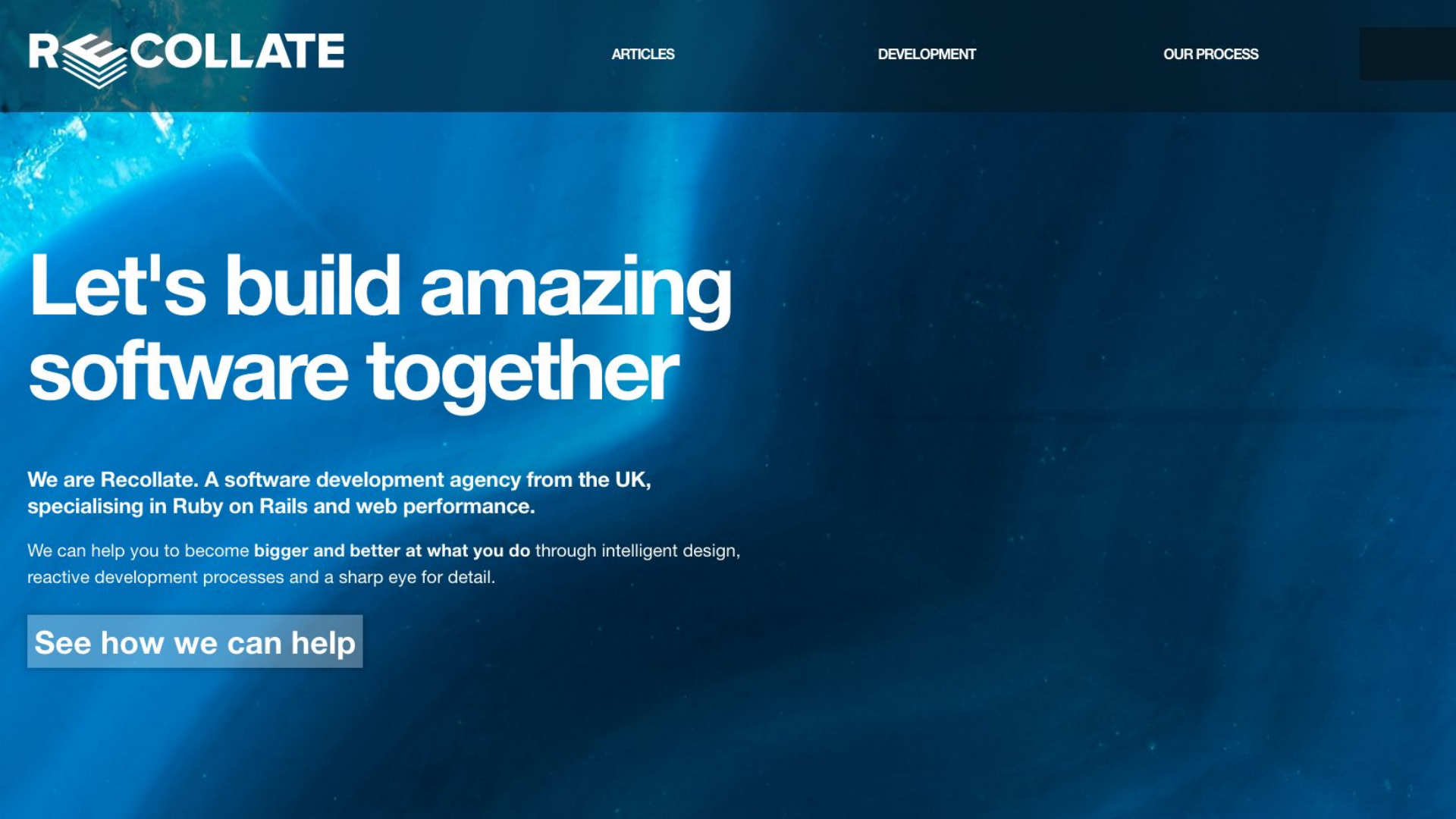
Status bar loading indicator

Loading pinwheels, then full site

rc8.io/mobile-perception

Who to follow

Lara Hogan (author/Etsy)	designingforperformance.com
Tobias Ahlin (Minecraft/Github/Spotify)	tobiasahlin.com
Philip Walton (Google)	philipwalton.com
Brad Frost	bradfrost.com
Cloudflare's Webinars	cloudflare.com/webinars


RECOLLATE
ARTICLES
DEVELOPMENT
OUR PROCESS
Let's build amazing software together
We are Recollate. A software development agency from the UK, specialising in Ruby on Rails and web performance.
We can help you to become bigger and better at what you do through intelligent design, reactive development processes and a sharp eye for detail.
See how we can help


recollate.com