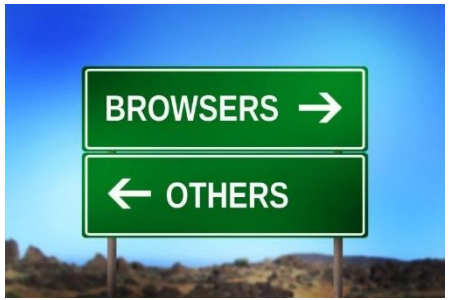
Considerations for non-browser HTTP clients

June 2017

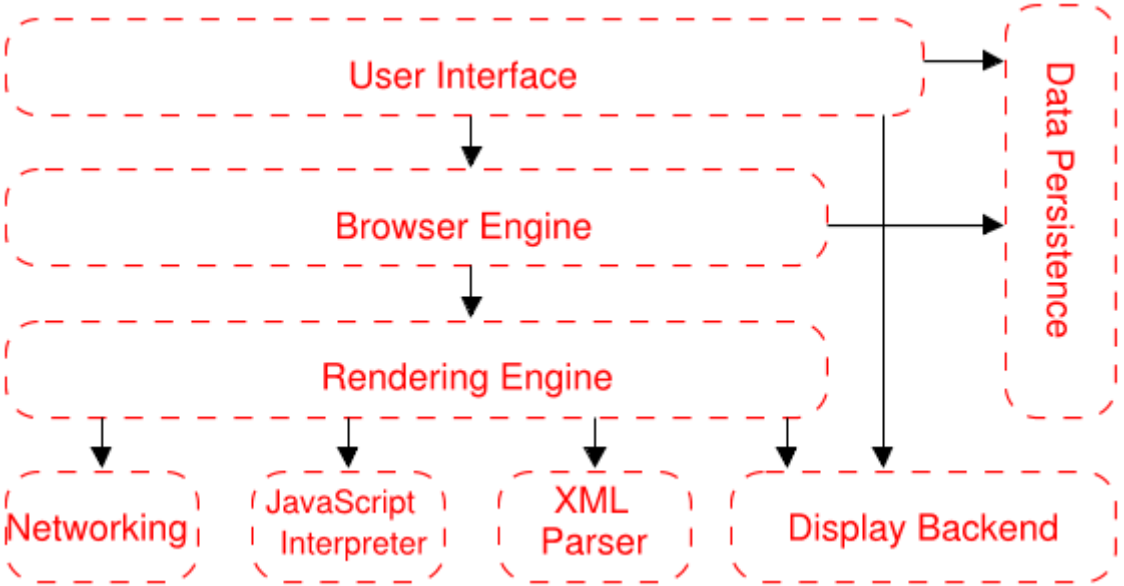
Lucas Pardue, R&D Project Engineer

What?

- What is a non-browser?
- What isn't a browser?
- What is browsing?



https://daniel.haxx.se/blog/2017/01/10/lesser-https-for-non-browsers/



A Reference Architecture for Web Browsers, Grosskurth and Godfrey, 2005

DATA

Performance Report

April 2017

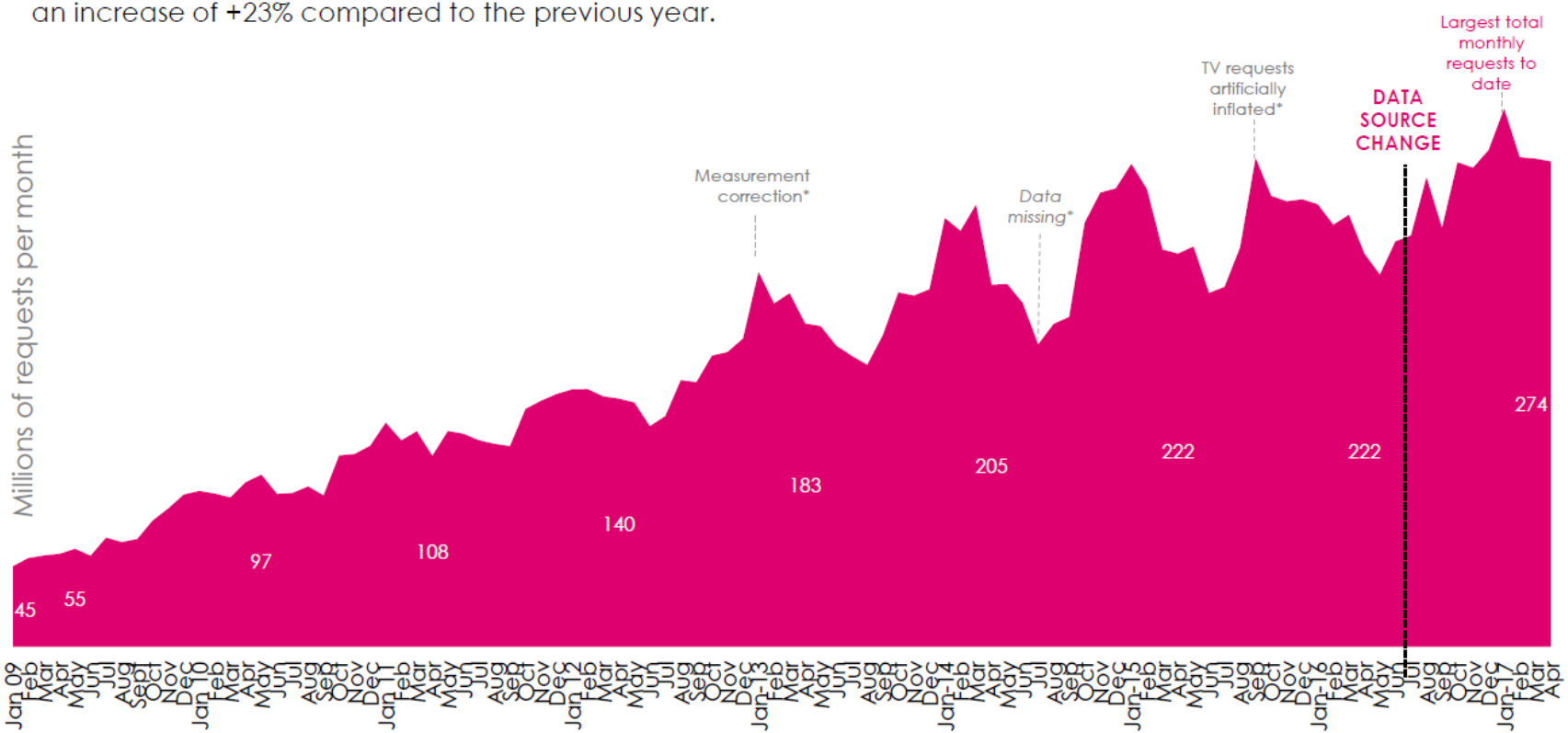
http://www.bbc.co.uk/mediacentre/search/?term=iplayer%20performance%20pack

Terminology

- **Requests** – the number of successful requests to stream or download a programme. We only count successful requests, where a stream or a download actually starts, rather than "clicks" which can be repeated if the user does not see an immediate reaction on the website. Requests are made up of two components:
  - **Stream** – click to play instantly
  - **Download** – save to your device to play later. We report download playback, rather than downloads, where possible.
- **Catch-up / on-demand** – programmes requested after they have gone out on traditional TV and are available on BBC iPlayer.
- **Live / simulcast** – streaming of live TV channels on the service, at exactly the same time as broadcast on traditional TV. Since May 2016, this data also includes webcasts of live events that are available through BBC iPlayer but not available on linear TV.

Monthly BBC iPlayer requests across all platforms, since 2009

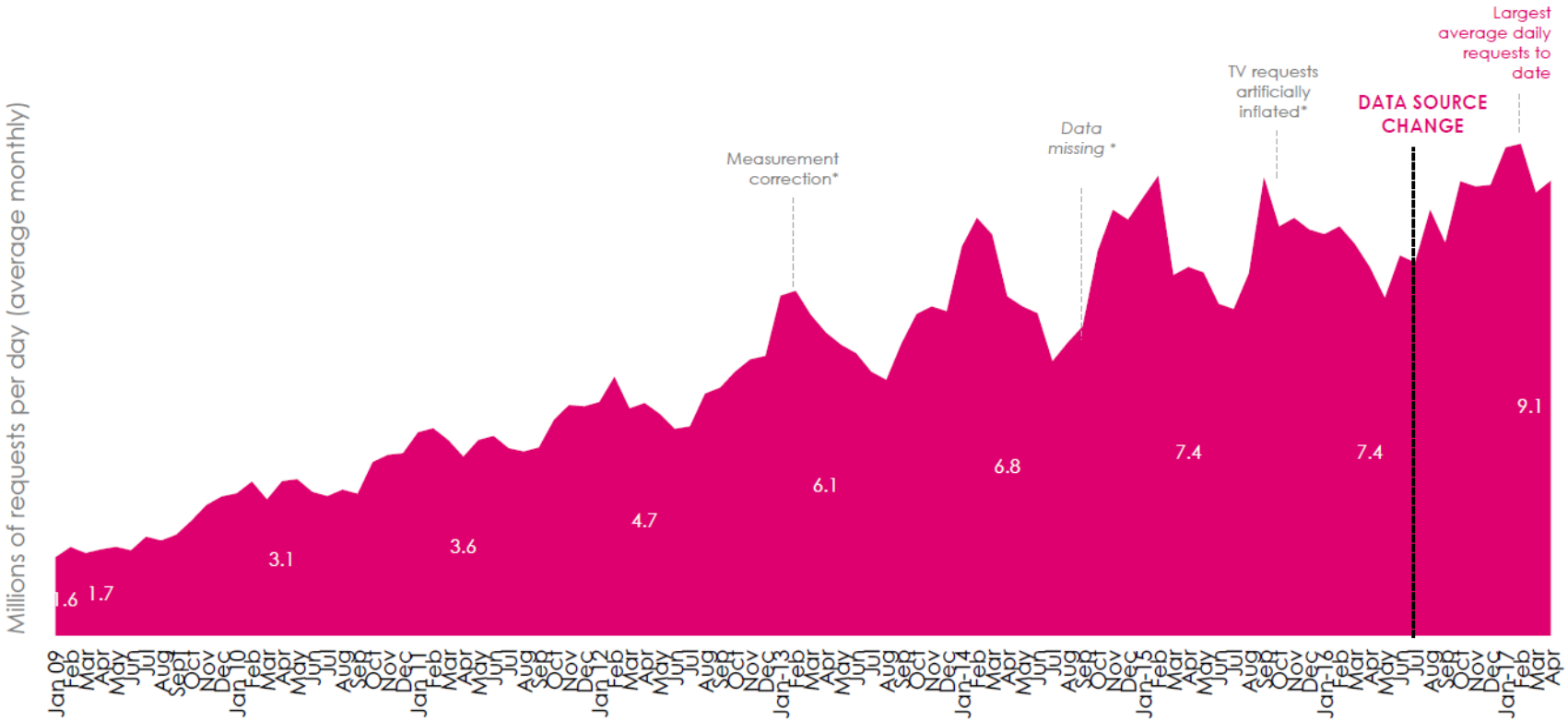
There were 274 million requests for TV programmes on BBC iPlayer in April, remaining stable compared to March, and an increase of +23% compared to the previous year.



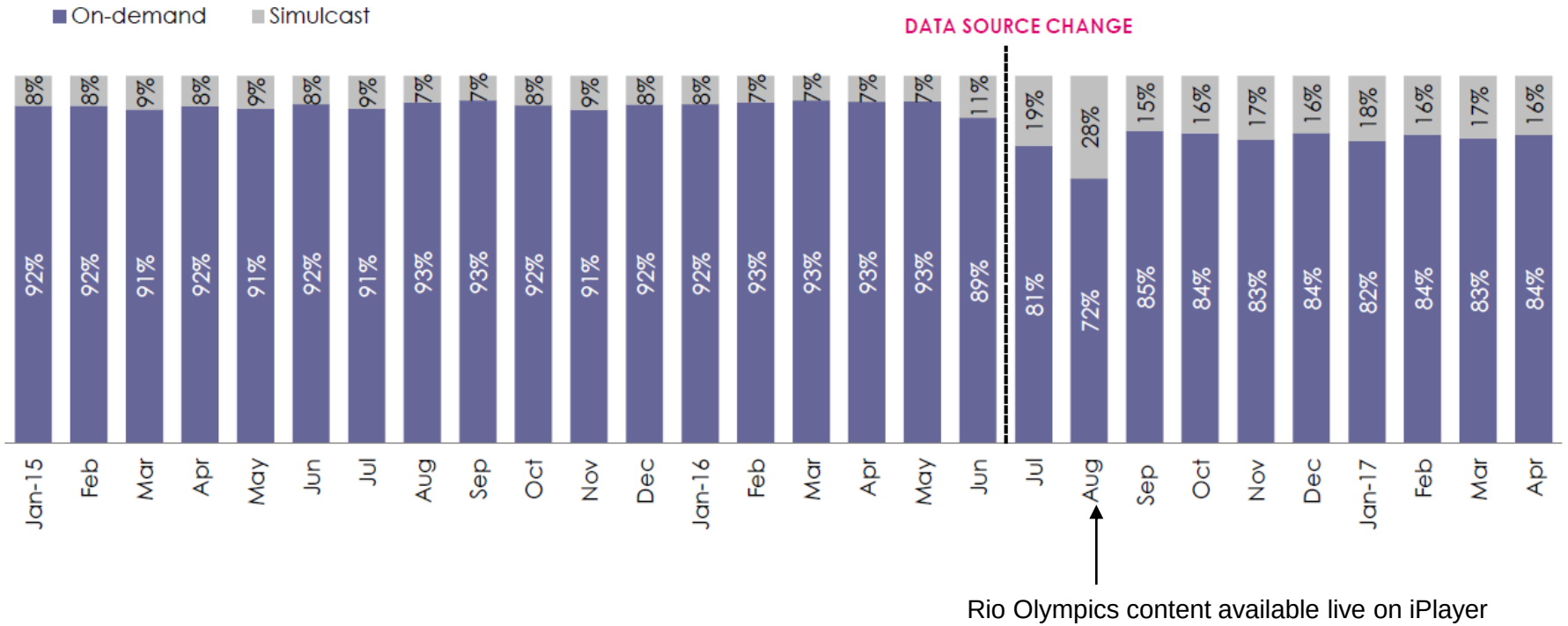
BBC | Research & Development

Average daily BBC iPlayer online requests

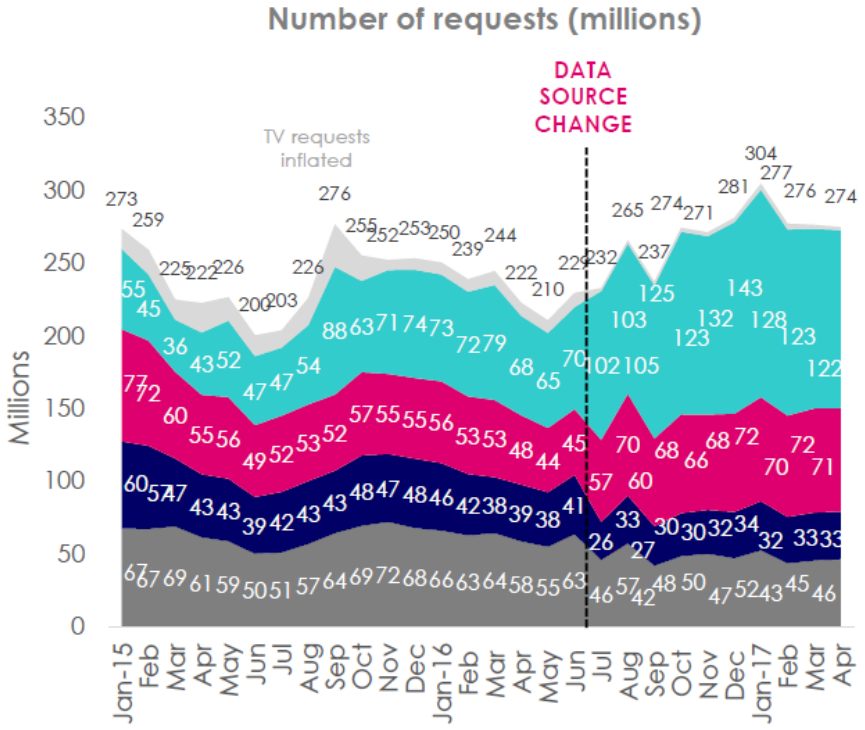
There was an increase in average daily requests for TV content in April, up +3% overall and reaching 9.1m.

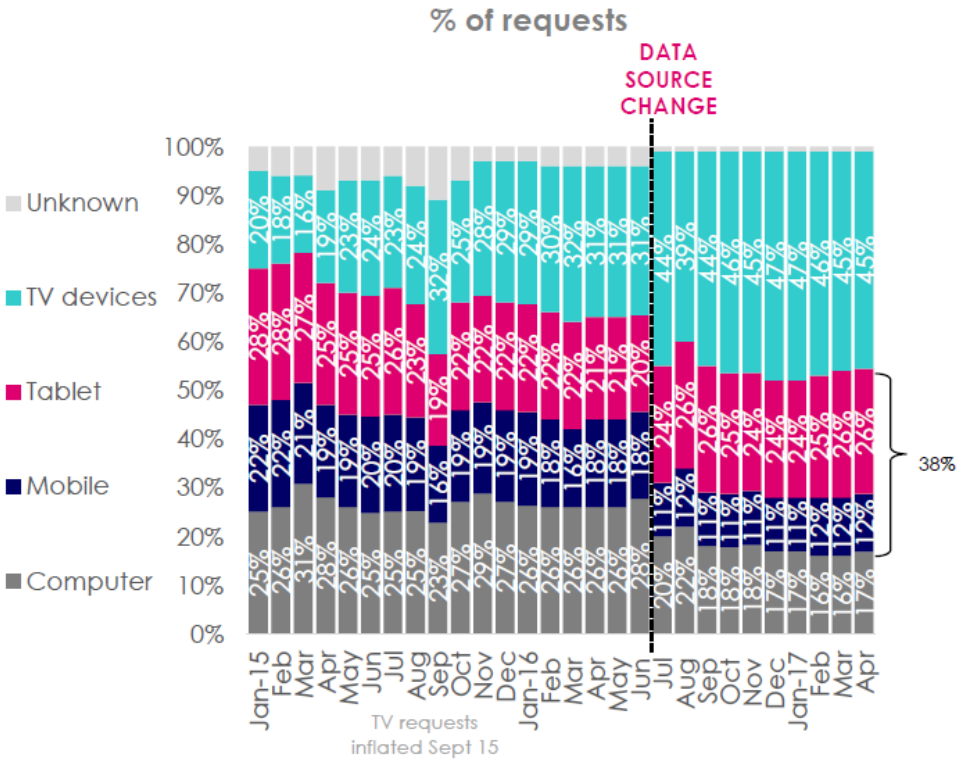


Live vs on-demand

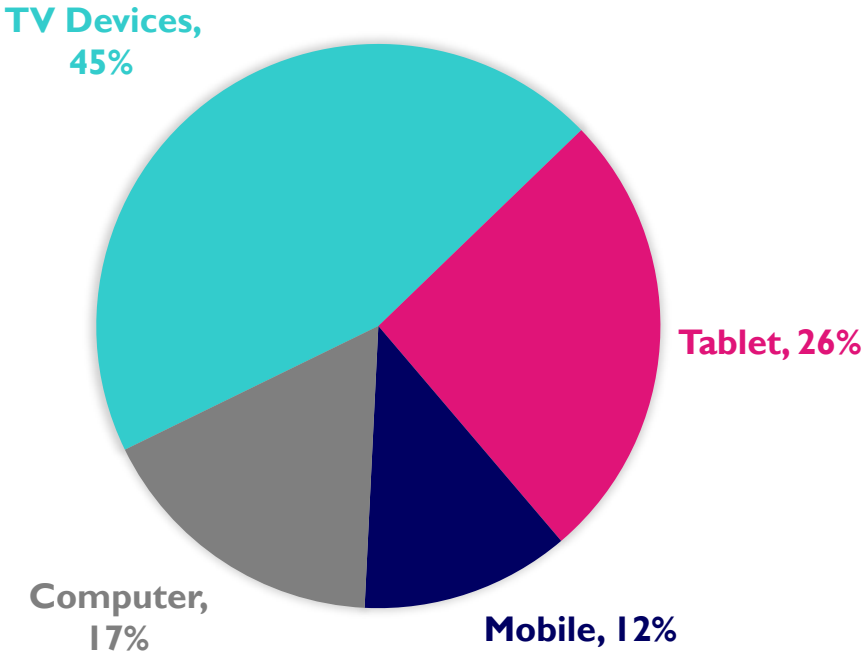


Requests by device type

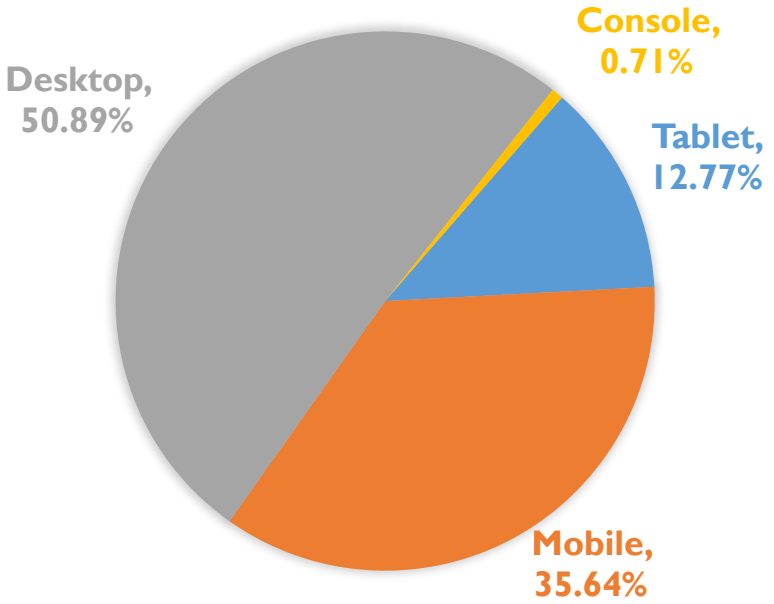




Device share April 2017



iPlayer

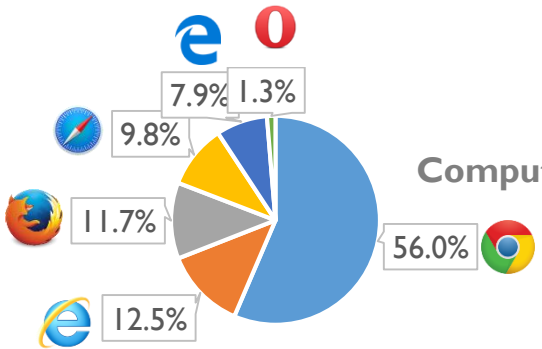


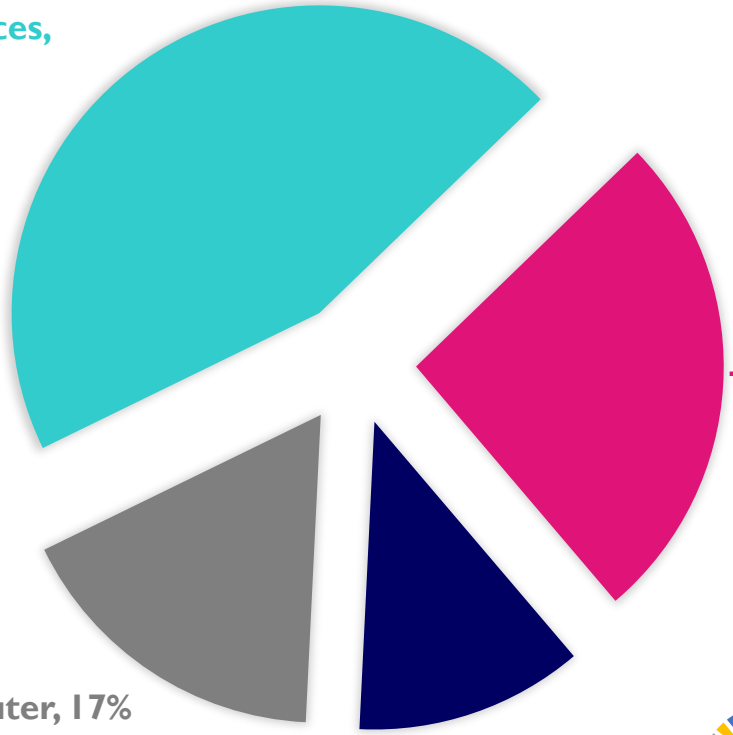
StatCounter Global Stats

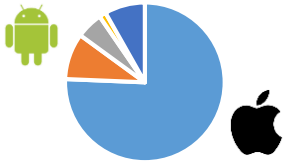
"stats are based on over 15 billion page views per month recorded across more than 2.5 million websites"

http://gs.statcounter.com/platform-market-share/all/united-kingdom/#monthly-201704-201704-bar

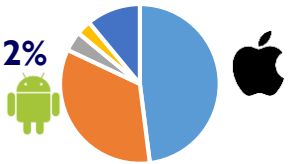
Requests by device type







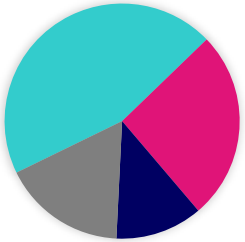
https://www.statista.com/statistics/487815/market-share-tablet-device-vendors-uk/



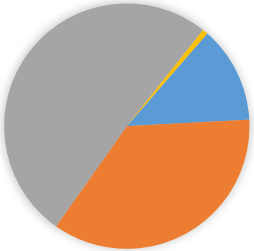
https://www.statista.com/statistics/387227/market-share-of-smartphone-manufacturers-in-the-uk/

http://gs.statcounter.com/browser-market-share/desktop/united-kingdom#monthly-201704-201705

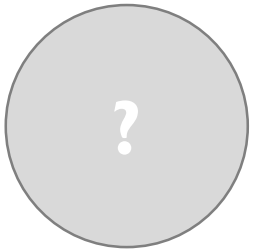
Classes of HTTP clients



Media
Streaming



Web
browsers



Caches

Scripts,
DevOps, etc

“Apps”
(downloads,
reporting, gaming, etc)

(micro) Service
APIs

IoT

Assistants

?

?

Characteristics of non-browsers HTTP clients

- Awareness and display of context
  - No URL bar
  - No “padlock”
  - No DOM
  - Same-origin policy, Cookies
- Handling errors
  - Exposing the error to the right layer
    - Even Fetch and MSE has this kind of problem
  - Rectifying the error
    - No user to reload page or tell an admin
- “Limited” devices
  - Monolithic images
  - Device support windows and maintenance schedules
  - Sometimes never connected
  - Sometimes contain multiple HTTP clients
- “Good enough” / “Means to an end”
  - Lack of awareness for considerations outside immediate problem domain
  - Select a client with minimal feature set required to get the job done
    - Security can be a barrier

Considerations for choosing an HTTP client

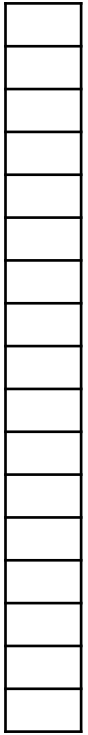
- Features (see next)
- Size
- Language
- Licence
- Integration
  - Into application code
    - Shell out
  - With other components or OS
    - i.e. is there a native capability or are there additional libraries being used
- Performance
- Privacy
  - Fingerprinting
- Security
- Maintenance, updates

Are these really any different to selecting any piece of software?

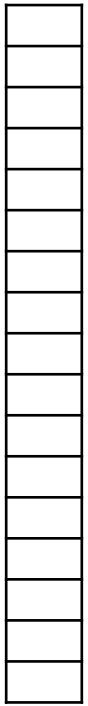
“Non-GUI” Features of HTTP clients

Low tech caniuse.com

- Methods
  - GET, POST, PUT, custom
- HTTP authentication
  - Basic, digest, NTLM, Negotiate
- HTTP Redirect
  - Limiting
- Caching
- Cookies
- Content parsing and recursion
  - JavaScript or JSON support
  - Recover and continue
- Compression/decompression
- Content-encoding, transport-encoding
- IDN, response charset
- Protocol evolution
  - HTTP/2, QUIC



- HTTP Alternative Services
- Happy eyeballs
- TLS*
  - Trusted CAs
  - Certificate transparency
  - HSTS
    - Preload
  - Session resumption
  - Cert revocation
  - Client certificates
  - SNI
  - Extended validation
- Proxy support
  - HTTP/HTTPS
  - SOCKS
- OAuth
- Etc, etc



* https://daniel.haxx.se/blog/2017/01/10/lesser-https-for-non-browsers/

A big, incomplete, list of non-browser HTTP clients

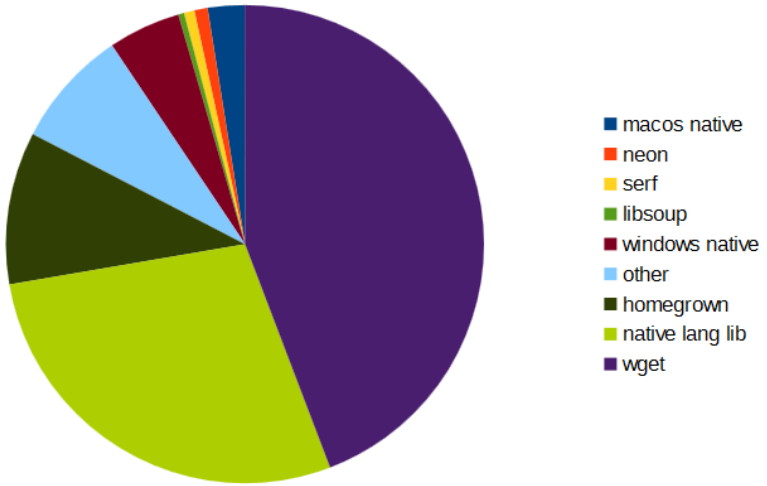
- libcurl
- libghttp
- libwww
- libferit
- neon
- libsoup
- mozilla netlib
- mozilla libxpnet (dead?)
- wget (extract code)
- libfetch
- HTTP Fetcher
- http-tiny
- wininet
- XMLHTTP Object
- QHttp
- GNU Common C++ Library
- gnetlibrary
- serf
- qDecoder
- HTTPClient
- Jakarta Commons HttpClient
- OkHttp
- node-fetch
- request
- Superagent
- NSURL

Curl user survey 2017

If you couldn't use libcurl, what would be your preferred alternative?

n = 448

The answer proportions to this answer remain very similar to past years.



Wget code remains the clear winner: 44.2%, native lang lib at 28.1% and homegrown at 10.3%

https://daniel.haxx.se/media/curl%20user%20poll%202017%20analysis.pdf

Case study: libsoup not featureful enough

- BBC R&D uses GStreamer for some projects, which uses libsoup by default.
- We have created a libcurl plugin to suit our feature requirements.
- This has been open sourced and is on GitHub
  - https://github.com/bbc/gst-curlhttpsrc
- The plugin is in the process of being brought into the official GStreamer collection.

Use case: HTTP/QUIC for non-browsers

- Main challenge is discovery
  - Alt-Svc is required in order to discover a QUIC server
  - A client that could talk QUIC only, cannot discover a QUIC server directly without TCP/TLS
  - See "HTTP/QUIC without Alt-Svc?" https://github.com/quicwg/base-drafts/issues/253
- AFAIK no non-browser HTTP clients support Alt-Svc
- "Good enough" solution
  - Parse the "Alt-Svc" header on responses and snoop for "hq"
- Proper solution should solve non-trivial aspects such as:
  - Presentation of URI to applications
  - Reasonable assurance of security
  - Background testing
  - Caching of Alt-Svc
- Where is the best place for this proper solution
  - Internal to each application
  - External in a reusable library (with lifecycle aspects still managed by the application)
    - Something more like TLS session ticket handling solutions?
  - As a system level daemon
    - Something more like DNS solutions?

Thanks!

June 2017
Lucas Pardue, R&D Project Engineer

@SimmerVigor
@BBCRD