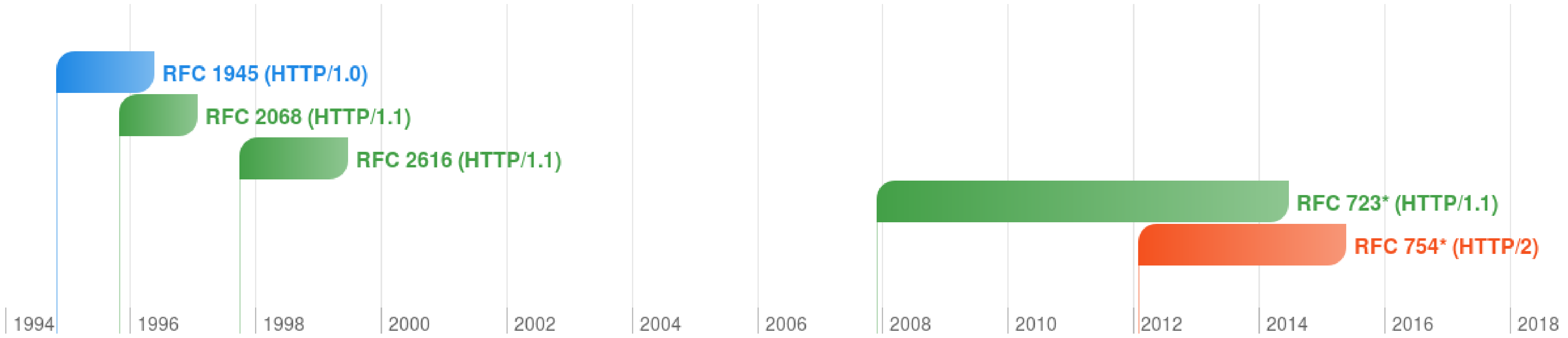
IETF 99 - HTTPtre

Julian Reschke, greenbytes

History of HTTP in the IETF



Observations:

- It took long to restart work on HTTP/1.1 (~8 years)
- It took long to finish the last revision of HTTP/1.1 (~ 6 years)
- There are **two** current HTTP specifications (HTTP/1.1 and HTTP/2)
    - ...not to mention HTTP over QUIC

Why and when to update

- RFCs are immutable documents
- We collect errata and occasionally revise
    - ...but the errata are incomplete and hard to discover (RFC-Editor???)
- If we start now, we won't be ready before 2018, which would be 4 years after publication of RFC 723x
- Eventually, the new RFC format will allow us to make the officially published versions more readable

Scope - the obvious

- Apply errata (see RFC 7230 errata etc)
- Update references (not too many, it seems)
- Resolve issues reported at https://github.com/httpwg/http11bis/issues, currently ~30
- Minimal changes to guide readers to HTTP/2 spec

Scope - the less obvious

Re-organize once more?

- Split information specific exclusively to HTTP/1.1 into a separate document and relabel everything else just "HTTP"
    - ...see draft-bishop-decomposing-http
- Recombine some or all of RFC7231..RFC7235 into a single document

Include stuff that should have been part of HTTP in the first place?

- Status 308 (RFC 7538)
- Authentication-Info and Proxy-Authentication-Info (RFC 7615)
- Client-Initiated Content-Encoding (RFC 7694)
- ...? Maybe draft-ietf-httpbis-rand-access-live?

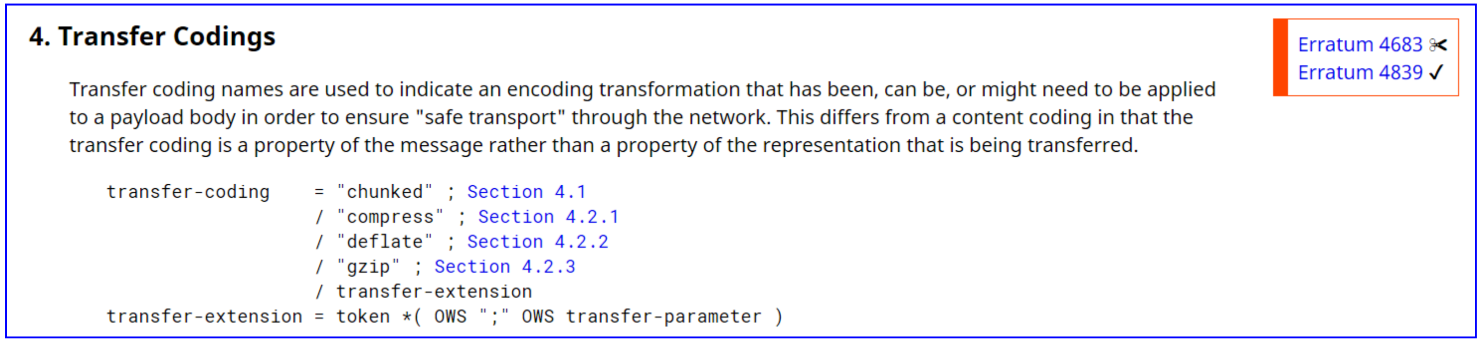
...advance to full standard? (~~may~~might conflict with other goals)

Next steps

- Motivate people to report their issues
- Discuss scope & timing
    - Relation to QUIC work (people overlap, potential effects for HTTP over QUIC)
    - Scope will affect how we do things, as potential re-organizations would need to be done very carefully

Shameless plug: annotating HTML versions of RFCs

We **can** inline some errata information already (in inofficial variants, that is):



4. Transfer Codings

Erratum 4683 ✂
Erratum 4839 ✓

Transfer coding names are used to indicate an encoding transformation that has been, can be, or might need to be applied to a payload body in order to ensure "safe transport" through the network. This differs from a content coding in that the transfer coding is a property of the message rather than a property of the representation that is being transferred.

```
    transfer-coding    = "chunked" ; Section 4.1
                       / "compress" ; Section 4.2.1
                       / "deflate" ; Section 4.2.2
                       / "gzip" ; Section 4.2.3
                       / transfer-extension
    transfer-extension = token *( OWS ";" OWS transfer-parameter )
```

...and we could extend that to information in Github issues.