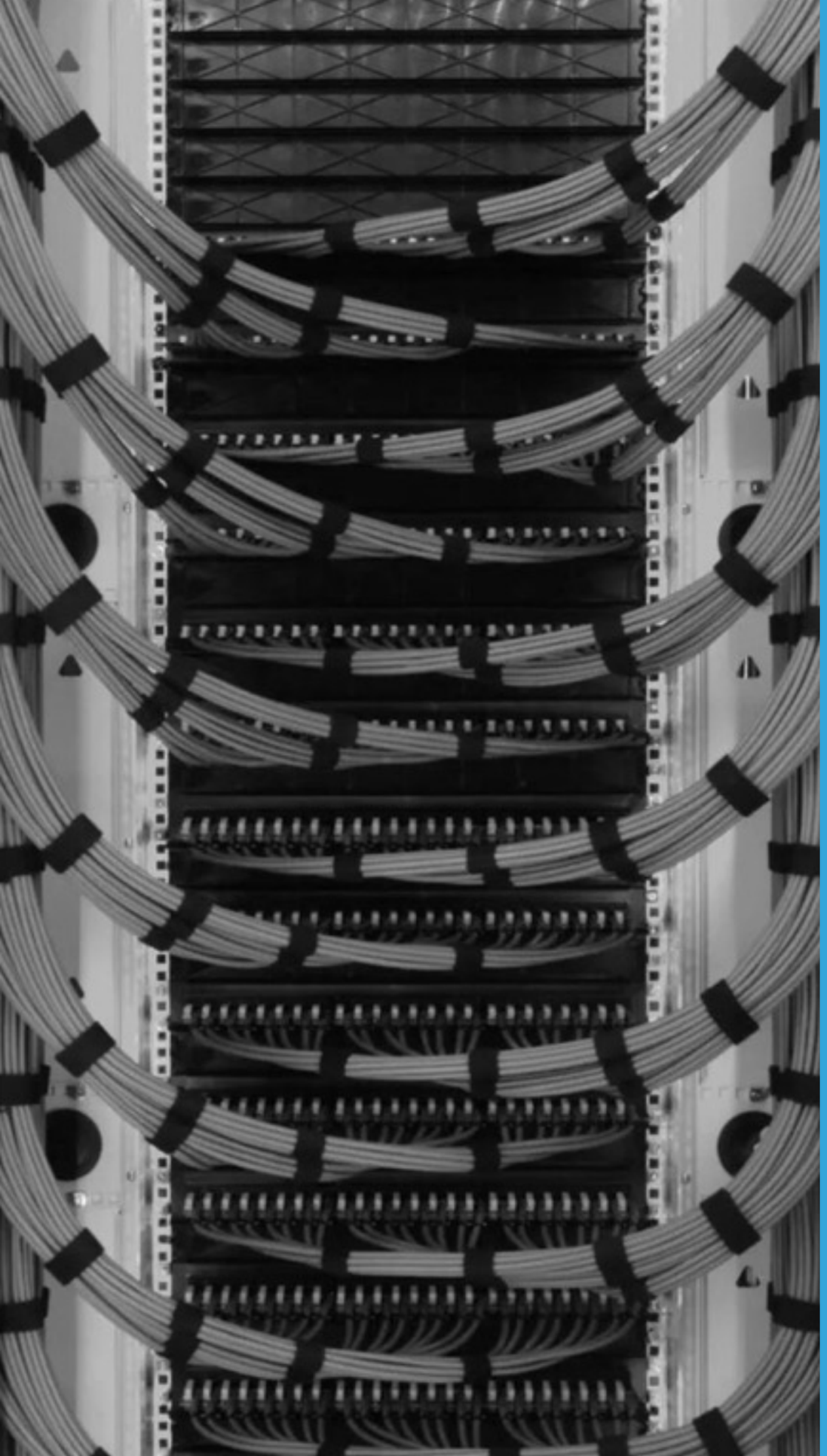
FRONTEND, RAILS, INFRASTRUCTURE

CACHING



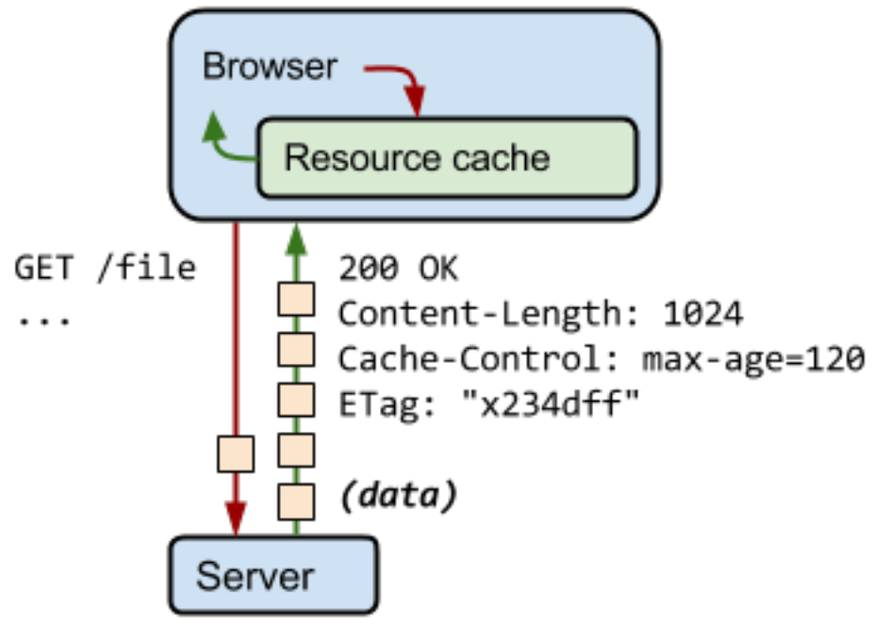
THERE ARE ONLY TWO HARD THINGS IN COMPUTER SCIENCE: CACHE INVALIDATION AND NAMING THINGS.

Phil Karlton

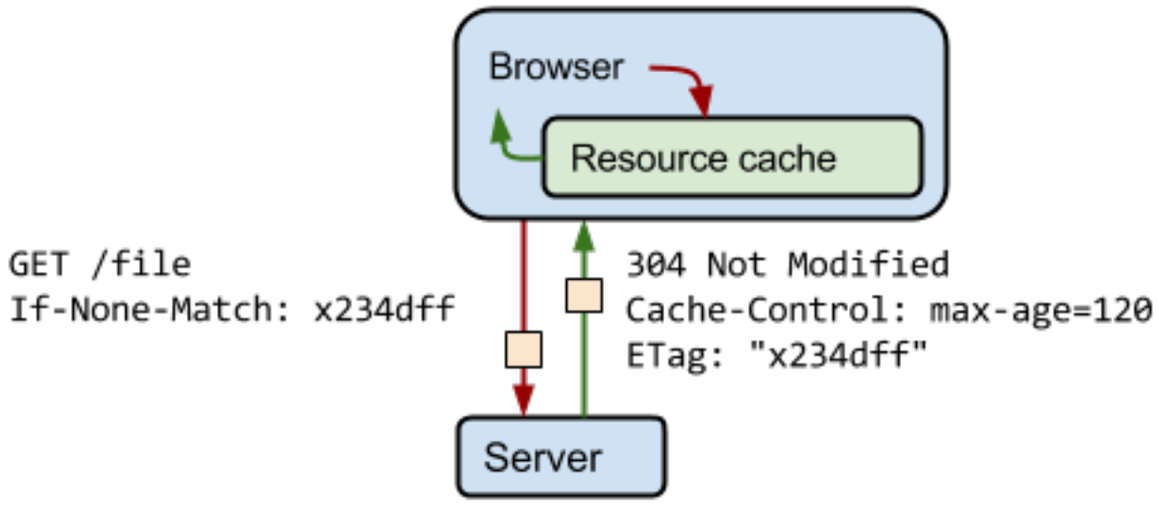
CACHE-HEADER

FRONTEND

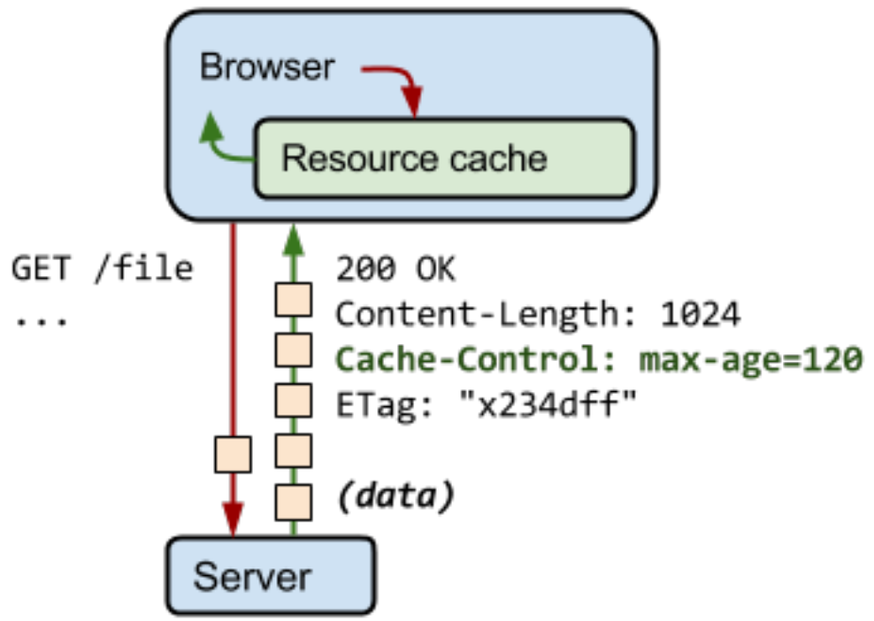
REQUEST/RESPONSE MIT CACHES



REQUEST/RESPONSE MIT CACHES: ETAG



REQUEST/RESPONSE MIT CACHES: MAX-AGE



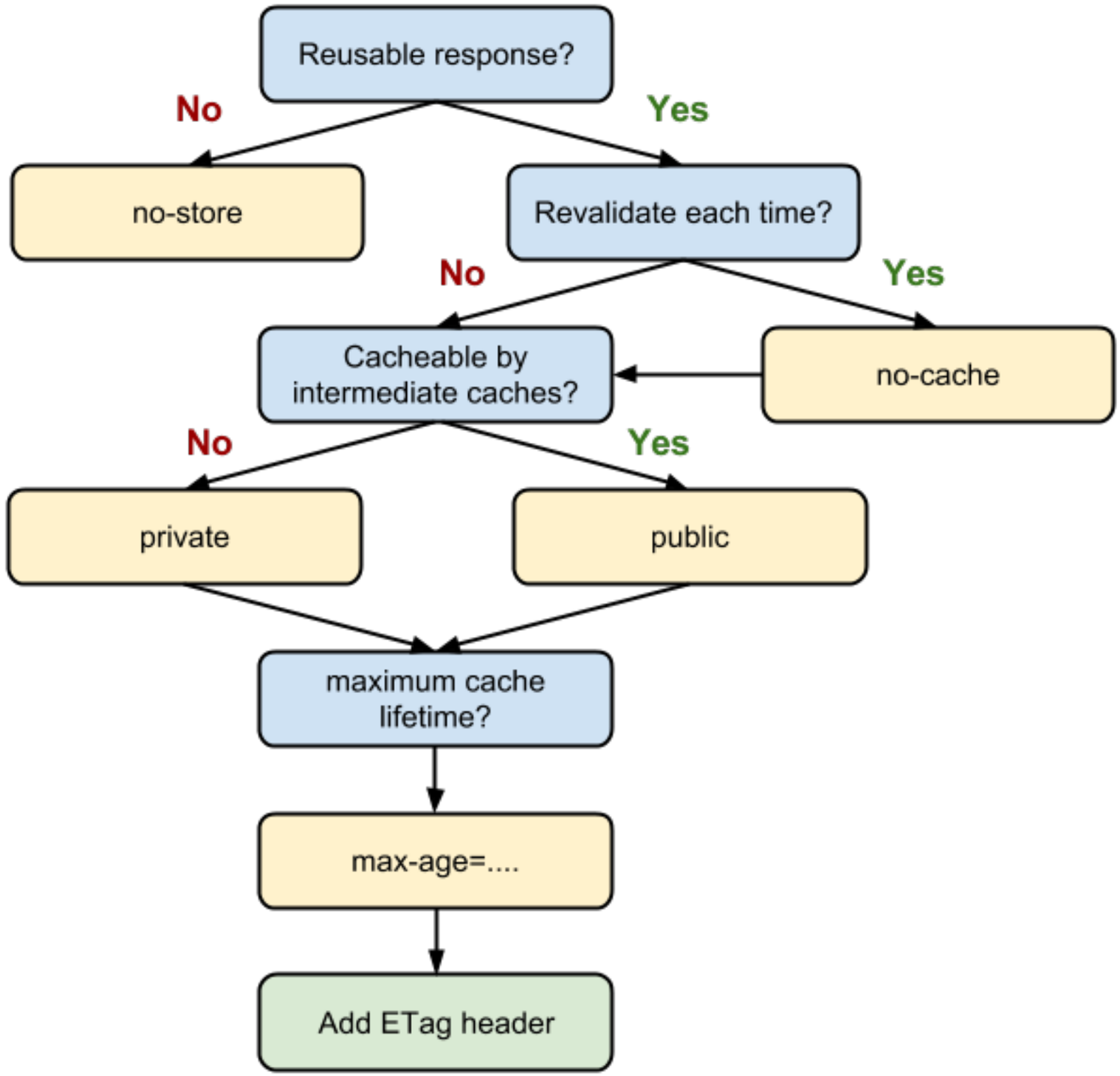
CACHE-HEADER

- cache-control: private, max-age=0, no-cache
  - private | public
  - no-cache | no-store
  - max-age
  - must-revalidate
- expires: Thu, 01 Dec 1983 20:00:00 GMT (obsolete)

CACHE-HEADER

- etag: einzigartiger Bezeichner für die angefragt Resource
  - z.B. ein hash des Contents
- vary: User-Agent
- pragma: no-cache (obsolete)

CACHING: FRONTEND



FRAGMENT, LOW-LEVEL, ...

RAILS-CACHING

FRAGMENT CACHING / RUSSIAN DOLL CACHING

- Generierung von Views ist eine der Hauptlasten auf Rails App-Servern
- Templates bzw. Blöcke werden gecacht, so dass nur geänderte Teile neu gerendert werden müssen.
- Invalidierung über cache_key und Template digest

```
<%# projects/show.html.erb %>
<% cache [ "v5", project ] do %>
  <p>All my todo lists:</p>
  <%= render project.todolists %>
<% end %>

<%# todolists/_todolist.html.erb %>
<% cache [ "v3", todolist ] do %>
  <p><%= todolist.name %>:</p>
  <%= render todolist.todos %>
<% end %>

<%# todos/_todo.html.erb %>
<% cache [ "v1", todo ] do %>
  <p><%= todo.name %></p>
<% end %>
```

LOW LEVEL CACHING

- Berechnungen oder Schnittstellenantworten mit längerer Gültigkeit zwischenspeichern.
- Rails.cache.fetch
- Gültigkeit kann angegeben werden
- manuelle Invalidierend möglich

CACHE-BACKENDS

- ActiveSupport::FileStore
- ActiveSupport::MemoryStore
- Memcache and dalli
- Redis and redis-store
- LRURedux

CONDITIONAL GET

- Rückgabe von 304, falls sich der Inhalt der Seite nicht geändert hat.
- Ausschließlich Übertragung des Headers

- stale?
- fresh_when
- Überprüfung ob ETag und/oder Last-Modified Header noch gültig sind.

VARNISH, NGINX

INFRASTRUCTURE

ALLGEMEINES

- Full Page Cache
- sehr schnell
- Steuerung üblicherweise über Cache-Header

BEDINGUNGEN

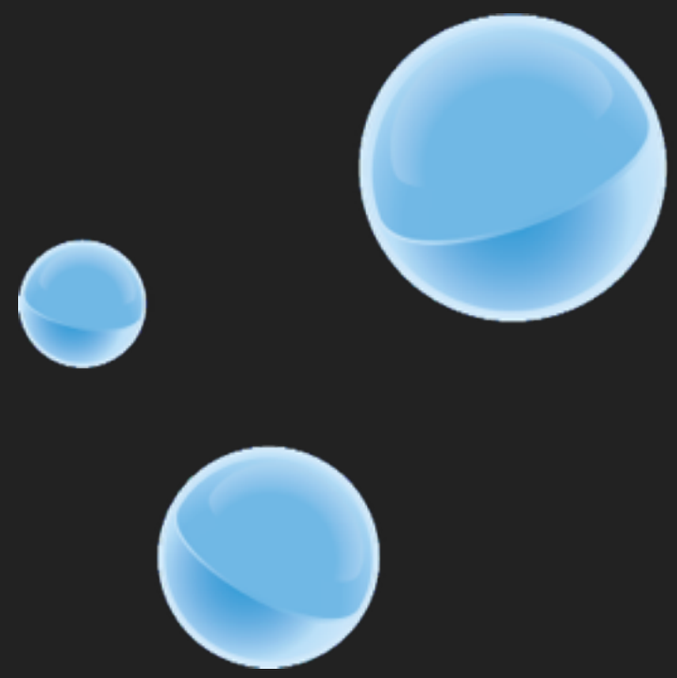
- Keine Cookies
- Keine dynamischen Inhalte
- oder Nachladen

NGINX

- Daten liegen auf HDD/SSD
- Konfiguration in NGINX config Dateien

VARNISH

- Daten werden im Hauptspeicher abgelegt.
- Entscheidung was im Speicher verbleibt liegt beim OS
- Thread-basierte Architektur
- Konfiguration über Varnish Configuration Language
- Lastenverteilung wird unterstützt



RESSOURCEN

- Frontend
  - https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching
  - https://www.mobify.com/blog/beginners-guide-to-http-cache-headers/
- Infrastructure
  - https://www.varnish-cache.org/
  - https://www.nginx.com/blog/nginx-caching-guide/
- Rails
  - http://guides.rubyonrails.org/caching_with_rails.html
  - https://www.nateberkopec.com/2015/07/15/the-complete-guide-to-rails-caching.html

FRAGEN?

VIELEN DANK FÜR EURE AUFMERKSAMKEIT