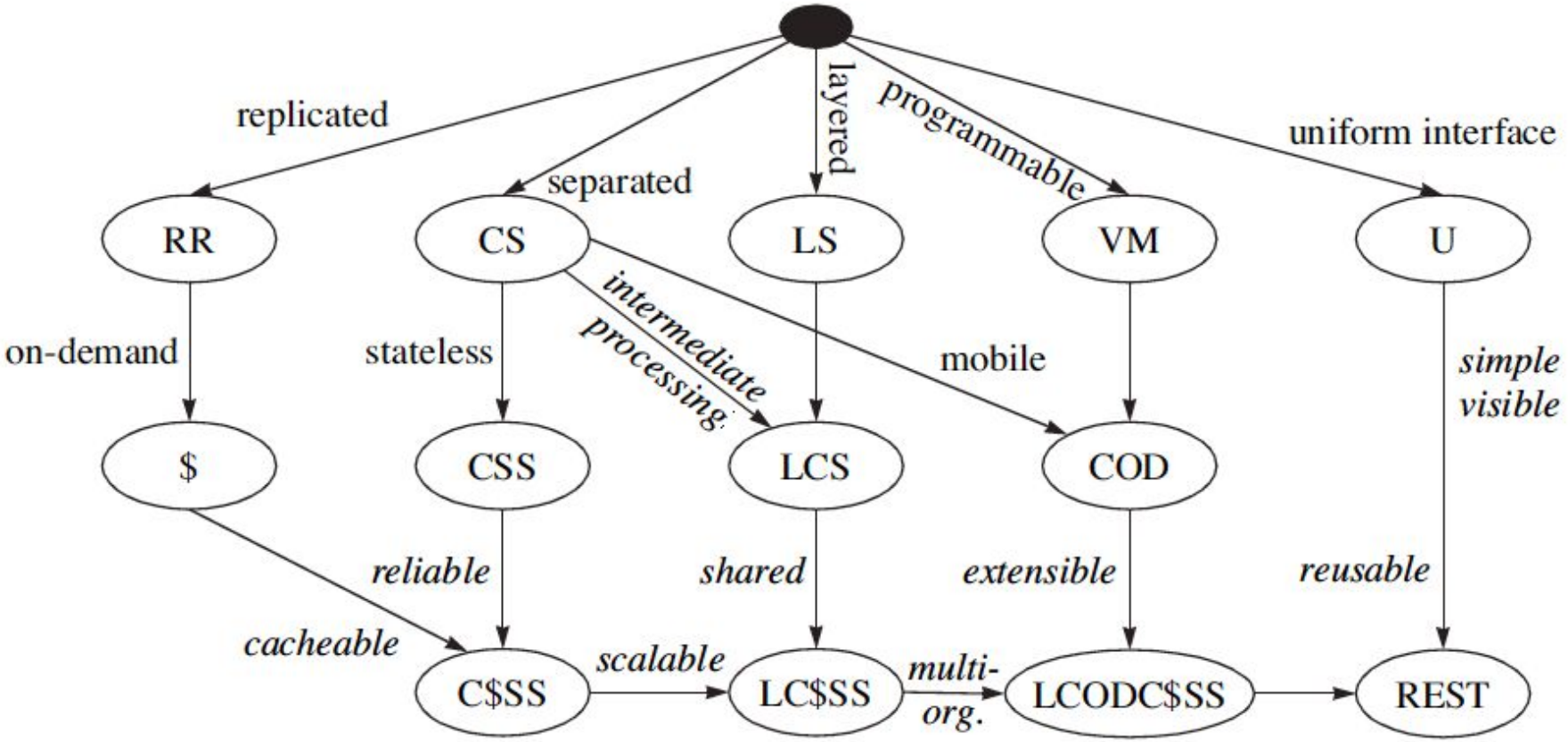
# REST Continued!

Oxford University
Software Engineering Programme
April 2021

# REST Derivation from Style Constraint



# Key concepts

- Client Server
- Caching
- Replicable
- Stateless
- Layered
- Uniform interface

# Cacheing

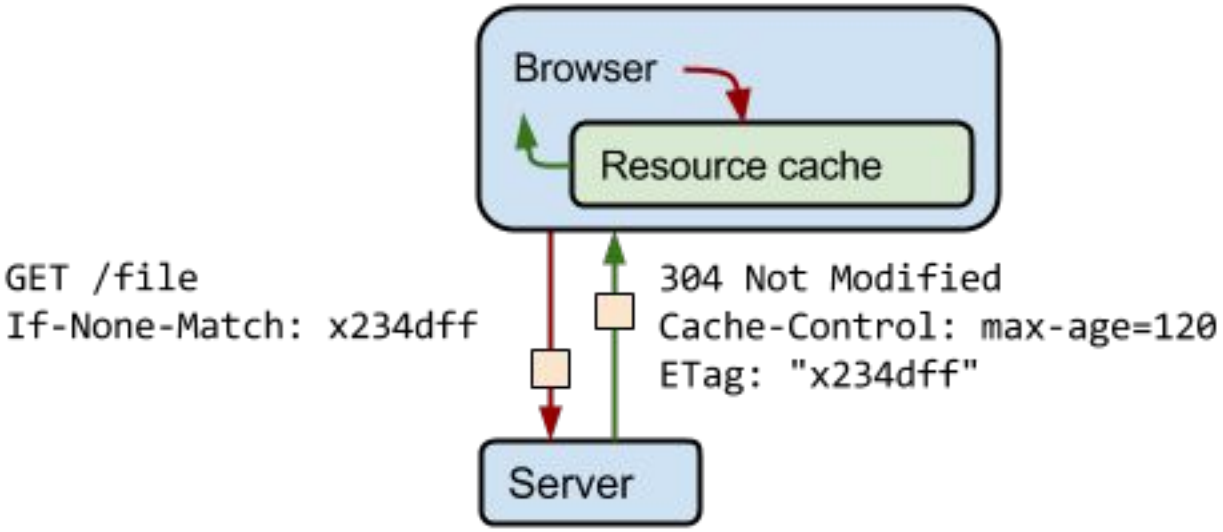
- Large scale network systems often rely on cacheing
  - Reduce traffic
  - Localised access
  - Reduced processing
    - **Akamai** and others make the web work effectively
- Caching relies on differentiating between cacheable and not cacheable traffic
  - Also understanding the lifetime and status of cacheable objects

# HTTP cacheing features

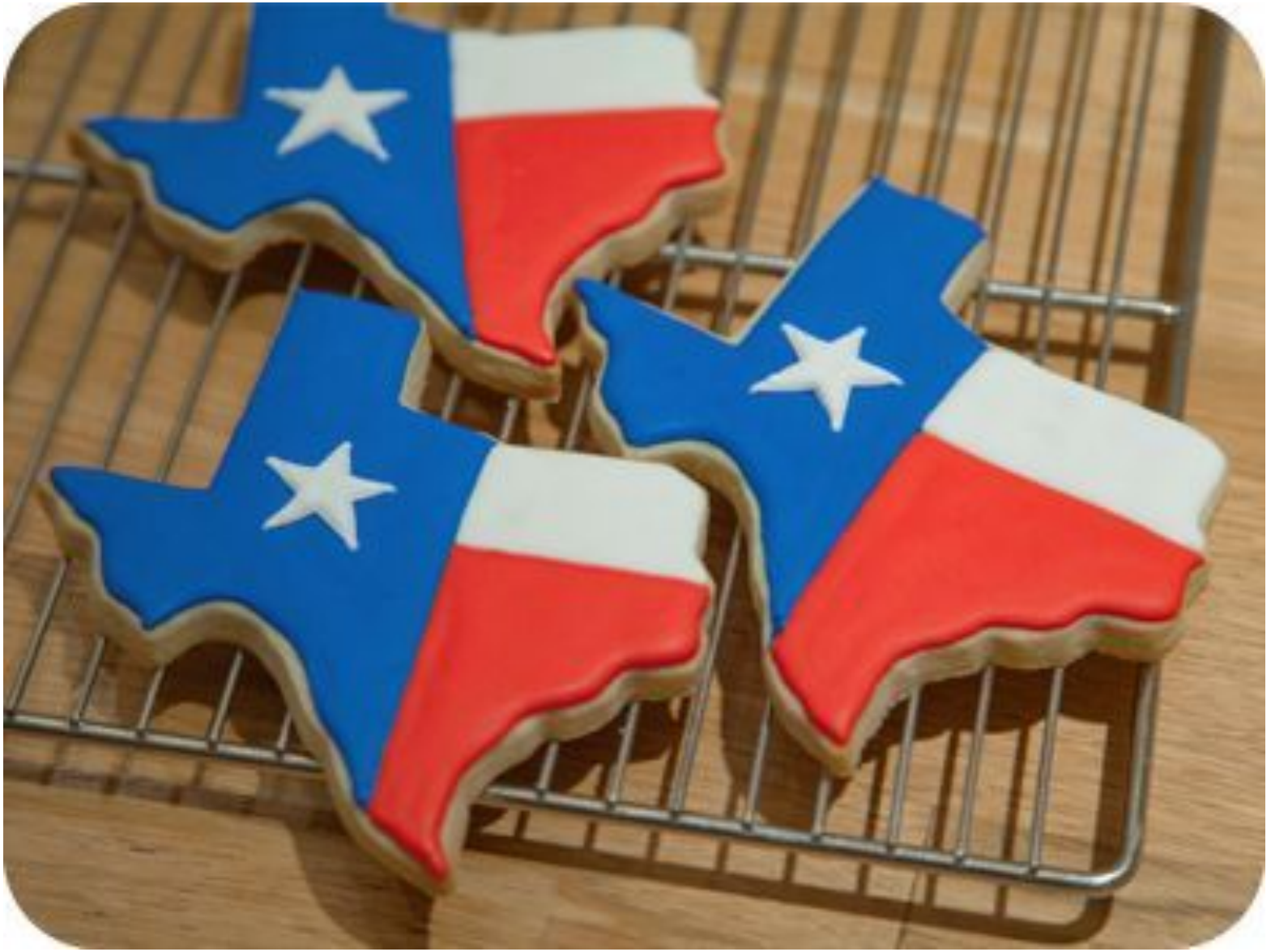
- Expires header
- Cache-control header
- If-modified-since / Not modified
- ETags (Entity Tags)
  - Uuids for content
  - Strong and Weak
- If None Match

- Unfortunately some websites are using Etags to track users instead of cookies!

# ETags




https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/images/http-cache-control.png



# Statelessness

- Of course there is state!
- The only question is where the state is kept.
- Traditional CS systems required the client and server to be kept in sync
- The web uses cookies

# Layered systems

- Reverse proxies
- Composable
  - E.g. my business process is a service that exposes a REST interface and co-ordinates other services
- Compare with web scraping

# Representation

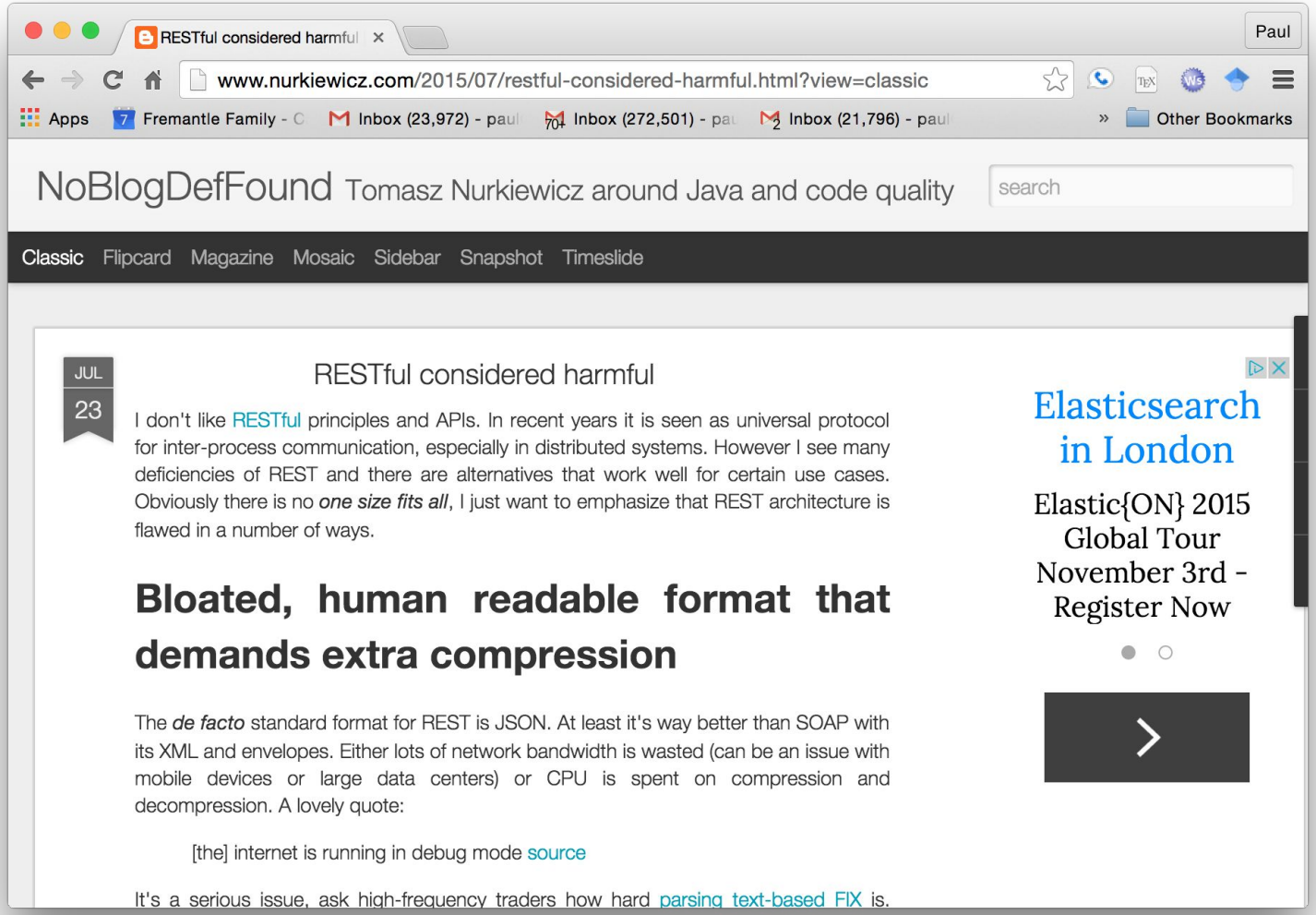
- State of resource captured and transferred between components
- Might be current or desired future state
- Represented as data plus metadata (name–value pairs)
- Metadata includes control data, media type
- The **Content-Type** of the resource should be useful and meaningful (self-description)
- One resource might have several representations
- Selected via separate URIs, or via content negotiation

# Uniform Interface

- A Uniform, Constrained Interface. When applying REST over HTTP, stick to the methods provided by the protocol
  - GET, POST, PUT, and DELETE.
- These should be used properly
  - GET should have no side effects or change on state
  - PUT should update the resource “in-place”

# Not everyone agrees:

http://www.nurkiewicz.com/2015/07/restful-considered-harmful.html



# Anti-REST concerns

- Bloated formats (equally applies to SOAP)
- Neither Schema nor Contract
- APIs and discovery instead of clear published machine-readable documentation
- No inbuilt batching, paging, sorting, etc
- CRUD only
- HTTP Status codes mixed with business replies
- Temporal Coupling
- Not clear enough what is REST and what isn't!
- Backwards compatibility

# Why REST Keeps Me Up At Night

News, Mobile

May. 15 2012 By Guest Author


*This guest post comes from Daniel Jacobson (@daniel_jacobson), director of engineering for the Netflix API. Prior to Netflix, Daniel ran application development for NPR where he created the NPR API, among other things. He is also the co-author of APIs: A Strategy Guide and a frequent contributor to ProgrammableWeb and the Netflix Tech Blog.*

With respect to Web APIs, the industry has clearly and emphatically landed on REST as the standard way to implement these services. And for good reason... REST, which is generally implemented as a one-size-fits-all solution, is an excellent choice for a most companies who wish to expose their content to third parties, mobile app developers, partners, internal teams, etc. There are many tomes about what REST is and how best to implement it, so I won't go into detail here. But if I were to sum up the value proposition to these companies of the traditional REST solution, I would describe it as:

> *REST APIs are excellent at handling requests in a generic way, establishing a set of rules that allow a large number of known and unknown developers to easily consume the services that the API offers.*

http://www.programmableweb.com/news/why-rest-keeps-me-night/2012/05/15

# Alternatives to REST

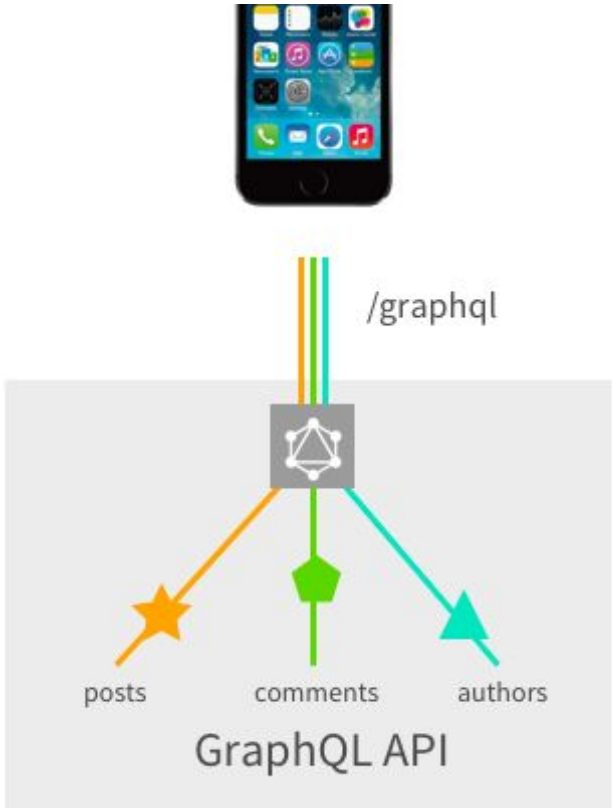
- GraphQL
- CQRS
- Event Sourcing
- Async and Events

# GraphQL


# GraphQL

- A Facebook developed language
- Supports getting many resources with a single call
- Allows the caller to specify what data is needed
- A single endpoint to access multiple resources with a well defined type system

# GraphQL vs REST


https://blog.apollographql.com/graphql-vs-rest-5d425123e34b

# Questions?