Demystifying use cases of REST, gRPC and GraphQL

Manaswini Das

REST in peace, SOAP

OCTOBER 15, 2010
IN TECH MUSINGS

REST is the new SOAP

Pakal De Bonchamp Follow
Dec 16, 2017 · 13 min read

JULY 24, 2017 / #GRAPHQL

REST APIs are REST-in-Peace APIs. Long Live GraphQL.

Samer Buna

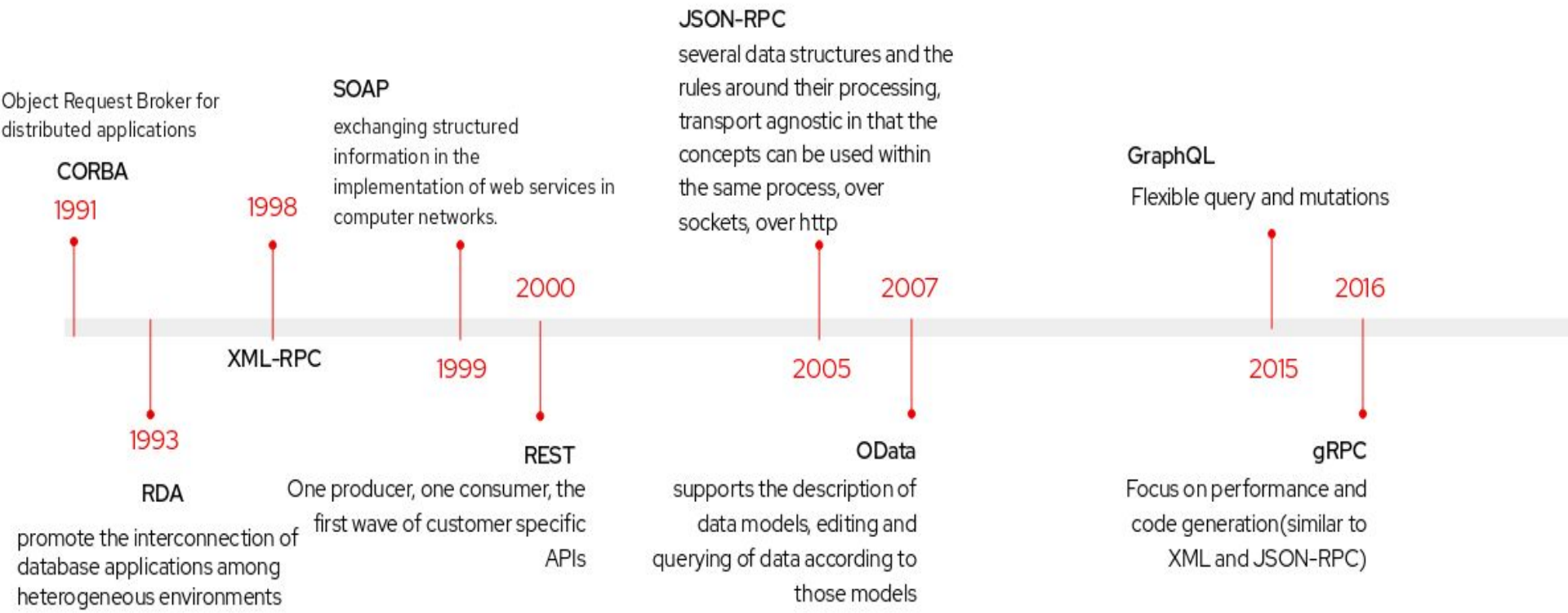
Sources:
https://www.pingdom.com/blog/rest-in-peace-soap/
https://www.freecodecamp.org/news/rest-apis-are-rest-in-peace-apis-long-live-graphql-d412e559d8e4/
https://medium.com/free-code-camp/rest-is-the-new-soap-97ff6c09896d

Agenda

- API timeline
- Constraints
- Busting myths
- Advantages and disadvantages
- Use cases

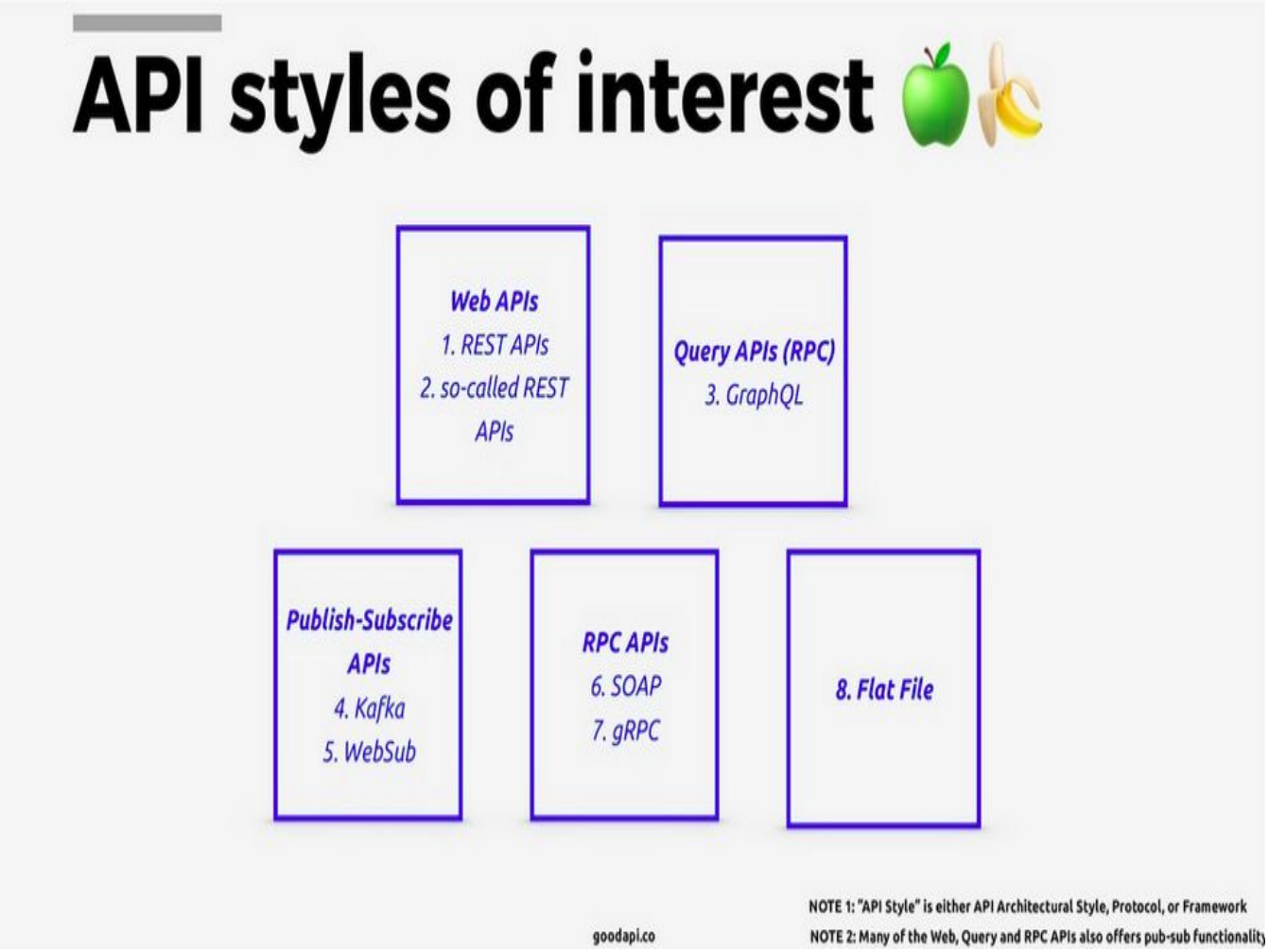
API timeline

Paradigm shifts



API styles

- Query
- Flat files
- Publish-subscribe
- RPC
- Web APIs



“Properties are induced by the set of constraints within an architecture”

Roy Fielding

Senior Principal Scientist, Adobe Systems

Constraints

- Business
- Technical
- Socio-technical

RDA + RPC = GraphQL

HTTP/2.x + protobuf = gRPC

“If an API is mostly actions, maybe it should be RPC. If an API is mostly CRUD and manipulates related data, it must be REST”

Phil Sturgeon

Software Engineer, WeWork

GraphQL breaks caching?

Over/under fetching and efficiency in REST

```
...?fields[resource-type]=field_name1, field_name2
```

Performance

- HTTP/1.x: cost of handshake was high
- HTTP/2.x removes need for compound documents(which ruined cacheability of requests)
- Server push creates new possibilities but profile use cases accordingly

Versioning?


"Be **conservative** in what you **send**, be **liberal** in what you **accept**"
Robustness Principle
(aka Postel's law)

Versioning

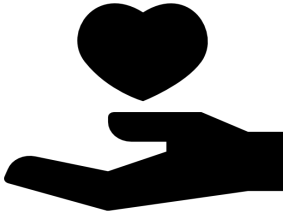
- Technique to handle breaking changes
- Prefer graceful evolution
- Not a substitute for communicating with users

Domain modelling is purely a REST concern

REST

- Standard method names, arguments and status codes
- Utilizes HTTP features
- Easy to maintain

- Big payloads
- Multiple HTTP roundtrips



- Best for APIs that expose CRUD like operations

RPC

- Easy to understand
- Lightweight payloads
- High performance

- Difficult discovery
- Limited standardization
- Leads to function explosion

- Best for APIs exposing several actions

GraphQL

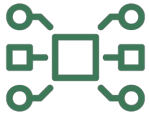
- Saves multiple round trips
- Smaller payload size

- Added complexity
- Difficult performance optimization
- Too complicated for simple API

- Best when you need querying flexibility

Specific use cases of REST, GraphQL and gRPC

Application web forms that involve simple CRUD



Composite web forms or microservices that involve schema stitching using multiple APIs, data source agnostic

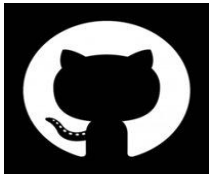
Native mobile BFF which involves single client, API is static and well documented, mostly single clients with real-time interaction

Further reading/viewing

- GraphQL, gRPC or REST- Resolving the API developers dilemma by Rob Crowley
- https://nordicapis.com/when-to-use-what-rest-graphql-webhooks-grpc/
- REST vs gRPC vs GraphQL - How do I pick the right API paradigm ?

Reach out to me

dasmanaswini10@gmail.com



manaswinidas

@ManaswiniDas4

Thank you