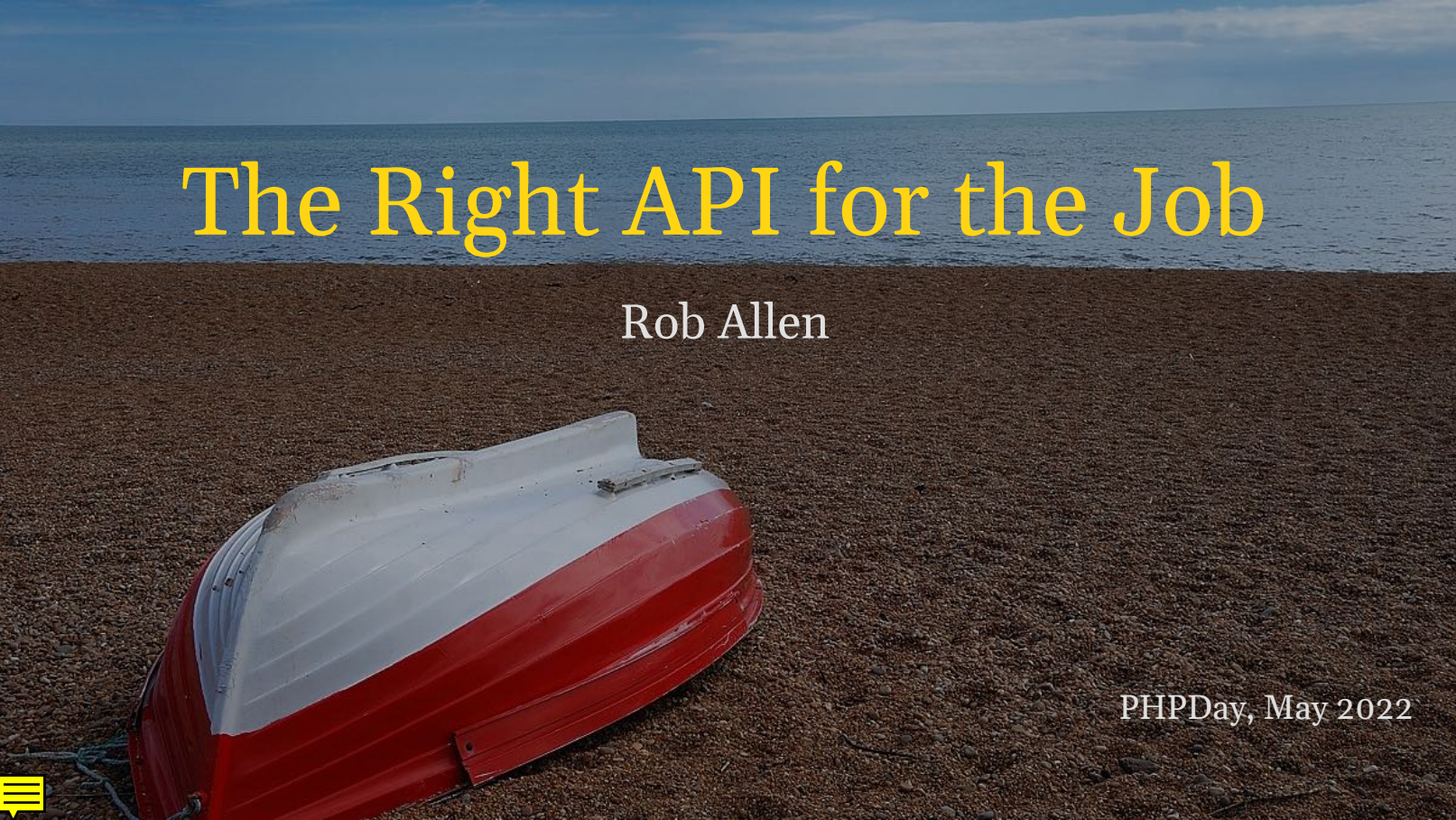
The Right API for the Job

Rob Allen



PHPDay, May 2022


Fit for Purpose

API Architecture

Rob Allen ~ @akrabat

APIs can be realised in any style
but, which makes the most sense?

RPC APIs

RPC APIs

- Call a function on a remote server

RPC APIs

- Call a function on a remote server
- Common implementations: JSON-RPC, SOAP, gRPC

RPC APIs

- Call a function on a remote server
- Common implementations: JSON-RPC, SOAP, gRPC
- Tends to require a schema (WSDL, ProtoBuf Defintion)

Ethereum JSON-RPC

Request:

```
POST / HTTP/1.1
Host: localhost:8545

{
  "jsonrpc":"2.0",
  "id":1,
  "method":"net_peerCount",
  "params":[]
}
```

Ethereum JSON-RPC

Response:

```
{
  "id":1,
  "jsonrpc": "2.0",
  "result": "0x2"
}
```

gRPC

Interact via PHP library:

```
$client = new RouteGuideClient('localhost:50051');

$p = new Routeguide\Point();
$p->setLatitude(409146138);
$p->setLongitude(-746188906);
list($feature, $status) = $client->GetFeature($p)->wait();
```

RESTful APIs

Rob Allen ~ @akrabat

RESTful APIs

- Operate on a representation of the state of a resource though HTTP verbs

RESTful APIs

- Operate on a representation of the state of a resource though HTTP verbs
- HTTP native

RESTful APIs

- Operate on a representation of the state of a resource though HTTP verbs
- HTTP native
- Uniform interface

RESTful APIs

- Operate on a representation of the state of a resource though HTTP verbs
- HTTP native
- Uniform interface
- Hypermedia controls

RESTful APIs

```
PUT /users/ba60c99fd3
Content-Type: application/json

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com"
}
```

RESTful APIs

```
PUT /users/ba60c99fd3
Content-Type: application/json

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com"
}
```

RESTful APIs

```
PUT /users/ba60c99fd3
Content-Type: application/json

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com"
}
```

RESTful APIs

```
HTTP/1.1 201 Created
ETag: dfb9f2ab35fe4d17bde2fb2b1cee88c1
Content-Type: application/json

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com",
    "_links": {
      "self": "https://api.example.com/user/ba60c99fd3"
    }
}
```

RESTful APIs

```
HTTP/1.1 201 Created
ETag: dfb9f2ab35fe4d17bde2fb2b1cee88c1
Content-Type: application/json

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com",
    "_links": {
      "self": "https://api.example.com/user/ba60c99fd3"
    }
}
```

RESTful APIs

```
HTTP/1.1 201 Created
ETag: dfb9f2ab35fe4d17bde2fb2b1cee88c1
Content-Type: application/json

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com",
    "_links": {
      "self": "https://api.example.com/user/ba60c99fd3"
    }
}
```

RESTful APIs

```
HTTP/1.1 201 Created
ETag: dfb9f2ab35fe4d17bde2fb2b1cee88c1
Content-Type: application/json

{
    "name": "Rob Allen"
    "email": "rob@akrabat.com",
    "_links": {
      "self": "https://api.example.com/user/ba60c99fd3"
    }
}
```

GraphQL APIs

Rob Allen ~ @akrabat

GraphQL APIs

- Retrieve only the data you need on consumer side

GraphQL APIs

- Retrieve only the data you need on consumer side
- Reduce the number of calls to retrieve data with embedded resources

GraphQL APIs

- Retrieve only the data you need on consumer side
- Reduce the number of calls to retrieve data with embedded resources
- Self-describing schema

Queries

```
query {
  author(name: "Ann McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title, datePublished
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Ann McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title, datePublished
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Ann McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title, datePublished
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Ann McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title, datePublished
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Ann McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title, datePublished
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Ann McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title, datePublished
        }
      }
    }
  }
}
```

Queries

```
query {
  author(name: "Ann McCaffrey") {
    id, name
    books(first: 5) {
      totalCount
      edges {
        node {
          id, title, datePublished
        }
      }
    }
  }
}
```

Queries


```
"data": {
  "author": {
    "id": "MXxBdXRob3J8ZjA",
    "name": "Ann McCaffrey",
    "books": {
      "totalCount": 6,
      "edges": [
        {
          "node": {
            "id": "MXxCb29rfGYwNzU",
            "title": "Dragonflight"
          }
        },
```

Queries


```
"data": {
  "author": {
    "id": "MXxBdXRob3J8ZjA",
    "name": "Ann McCaffrey",
    "books": {
      "totalCount": 6,
      "edges": [
        {
          "node": {
            "id": "MXxCb29rfGYwNzU",
            "title": "Dragonflight"
          }
        },
```

Queries


```
"data": {
  "author": {
    "id": "MXxBdXRob3J8ZjA",
    "name": "Ann McCaffrey",
    "books": {
      "totalCount": 6,
      "edges": [
        {
          "node": {
            "id": "MXxCb29rfGYwNzU",
            "title": "Dragonflight"
          }
        },
```

Queries


```
"data": {
  "author": {
    "id": "MXxBdXRob3J8ZjA",
    "name": "Ann McCaffrey",
    "books": {
      "totalCount": 6,
      "edges": [
        {
          "node": {
            "id": "MXxCb29rfGYwNzU",
            "title": "Dragonflight"
          }
        },
```


Mutations

```
mutation {
  createAuthor(
      name:"Mary Shelley", dateOfBirth: "1797-08-30"
  ) {
    returning {
      id, name
    }
  }
}
```

Mutations

```
mutation {
  createAuthor(
      name:"Mary Shelley", dateOfBirth: "1797-08-30"
  ) {
    returning {
      id, name
    }
  }
}
```

Mutations

```
mutation {
  createAuthor(
      name:"Mary Shelley", dateOfBirth: "1797-08-30"
  ) {
    returning {
      id, name
    }
  }
}
```

Mutations

```
mutation {
  createAuthor(
      name:"Mary Shelley", dateOfBirth: "1797-08-30"
  ) {
    returning {
      id, name
    }
  }
}
```

Mutations

Response:

```
"data": {
  "createAuthor": {
    "returning": [
      {
        "id": "e3388cbea4e840a",
        "name": "Mary Shelly",
      }
    ]
  }
}
```


Which to pick?
Rob Allen ~ @akrabat
Eduardo Llanquileo


Lamborghini or Ferrari?

Lamborghini or Truck?

Considerations

- What is it to be used for?
- Response customisation requirements
- HTTP interoperability requirements
- Binary protocol?

Response customisation

- GraphQL is a query-first language
- REST tends towards less customisation
- With RPC you get what you're given!

(None will fix your database layer's ability to efficiently retreive the data requested!)

Performance

- REST and RPC puts server performance first
- GraphQL puts client performance first

Caching

- GraphQL and RPC can only cache at application layer
- REST can additionally cache at HTTP layer

Data Transfer

GraphQL:

```
query {
  avatar(userId: "1234")
}

{
  "data": {
    "avatar": "(base64 data)"
    "format": "image/jpeg"
  }
}}
```

RPC:

```
POST /api
{
  "method": "getAvatar",
  "userId": "1234"
}

{
  "result": "(base64 data)"
}
```

Data Transfer

REST:

```
GET /user/1234/avatar
Accept: image/jpeg

HTTP/1.1 200 OK
{jpg image data}
```

REST:

```
GET /user/1234/avatar
Accept: application/json

HTTP/1.1 200 OK
{"data": "(base64 data)"}
```

Versioning

- RPC, GraphQL and REST can all version via evolution as easily as each other

Versioning

- RPC, GraphQL and REST can all version via evolution as easily as each other
- GraphQL is very good for deprecation of specific fields

Design considerations

It's *always* hard!

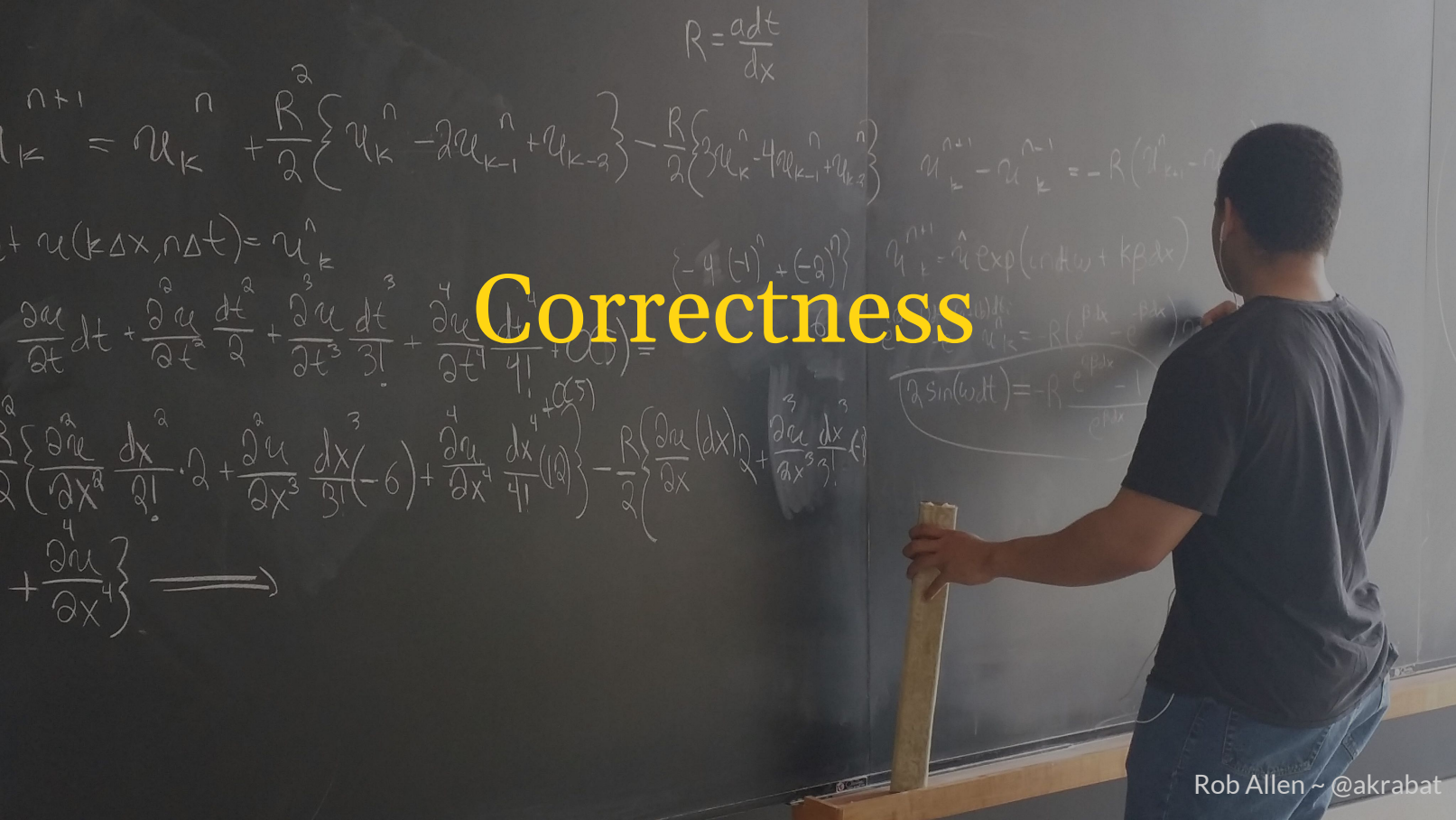
Design considerations

It's *always* hard!

It's your choice

Developer Experience

Rob Allen ~ @akrabat



Rob Allen ~ @akrabat

Correctness

RPC: Functions!

Correctness

RPC: Functions!
REST: HTTP matters!

Correctness

RPC: Functions!
REST: HTTP matters!
GraphQL: Think in terms of relationships!

Correctness

RPC: Functions!
REST: HTTP matters!
GraphQL: Think in terms of relationships!


Errors
FO3409OA01
ADAC
EURO NCAP
Rob Allen ~ @akrabat

Errors

Error representations must be first class citizens

Errors

Error representations must be first class citizens


Documentation
Rob Allen ~ @akrabat

Documentation

- API Reference

Documentation

- API Reference
- Tutorials

To sum up

If you suck at providing a REST API,
you will suck at providing a GraphQL API

Arnaud Lauret, API Handyman


Thank you!
https://joind.in/talk/8cdd9
Rob Allen ~ @akrabat

Photo credits

- Architecture: https://www.flickr.com/photos/shawnstilwell/4335732627
- Choose Pill: https://www.flickr.com/photos/eclib/4905907267
- Lamborghini & Ferrari: https://akrab.at/3w0yFmg
- Lamborghini & Truck: https://akrab.at/3F4kAZk
- '50s Computer: https://www.flickr.com/photos/9479603@N02/49755349401
- Blackboard: https://www.flickr.com/photos/bryanalexander/17182506391
- Crash Test: https://www.flickr.com/photos/astrablog/4133302216