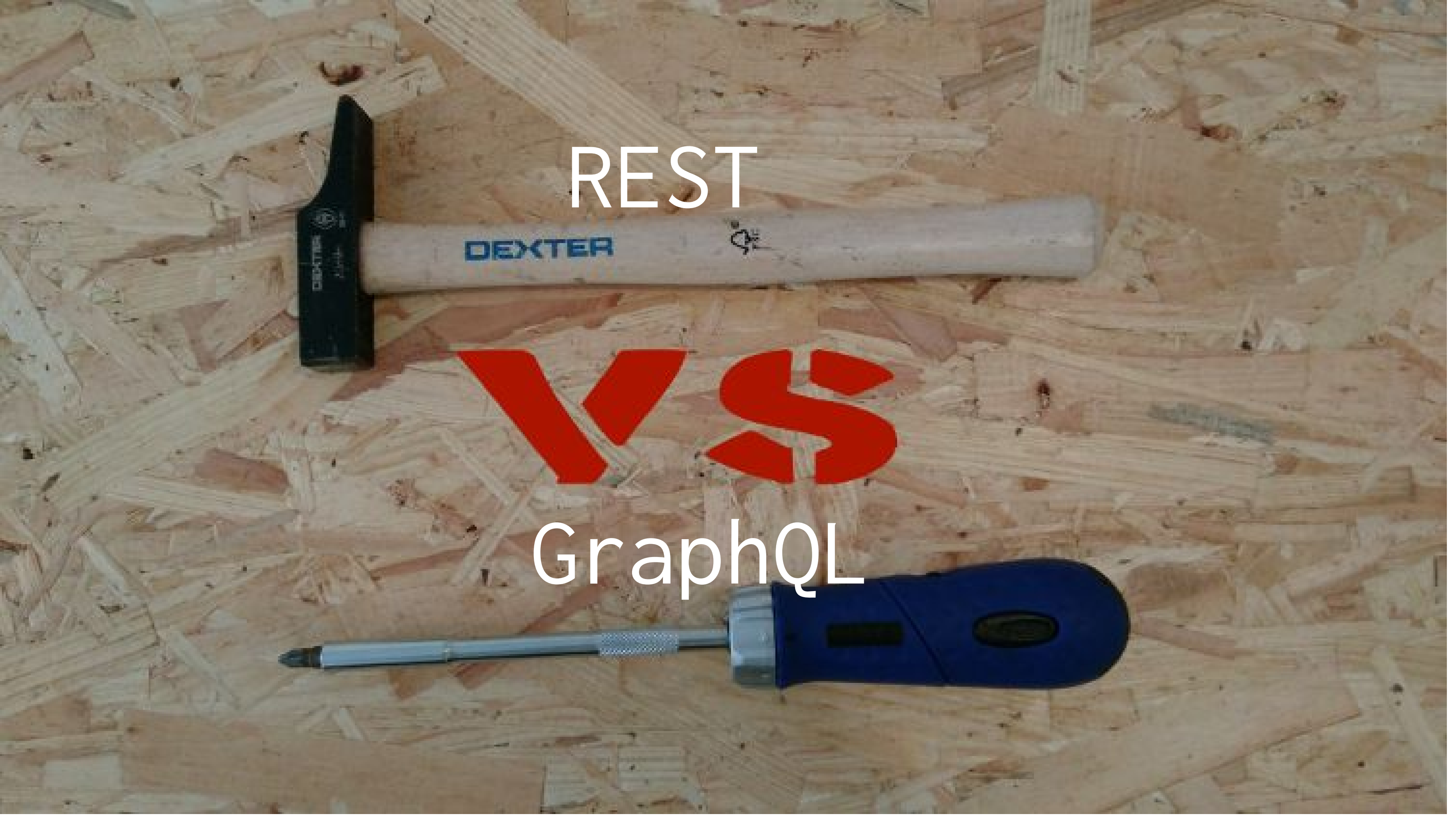
50 shades of **GraphQL**

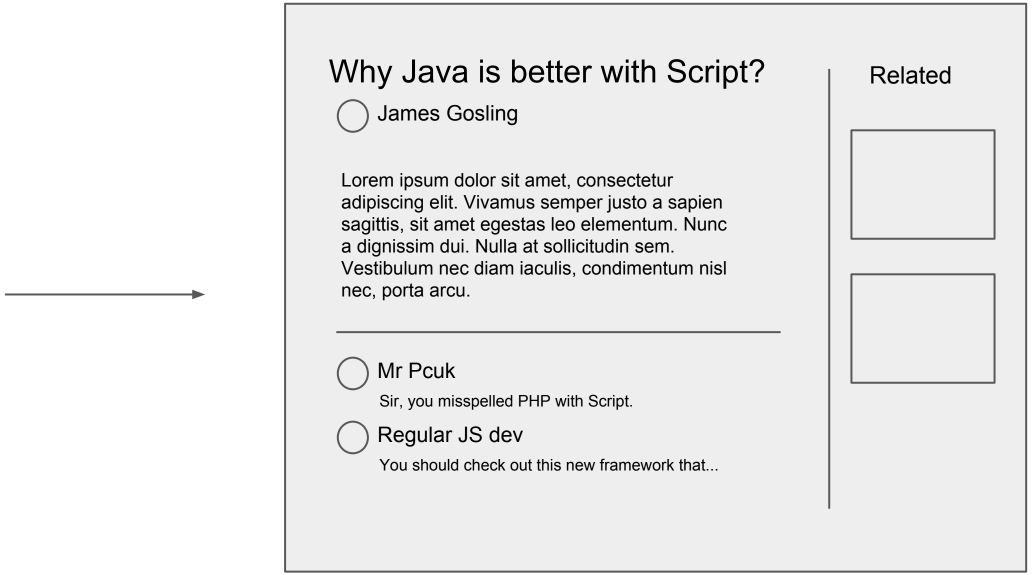
ACT I

Scene 1: REST enters the room, shouting


REST
VS
GraphQL
DEXTER

REST

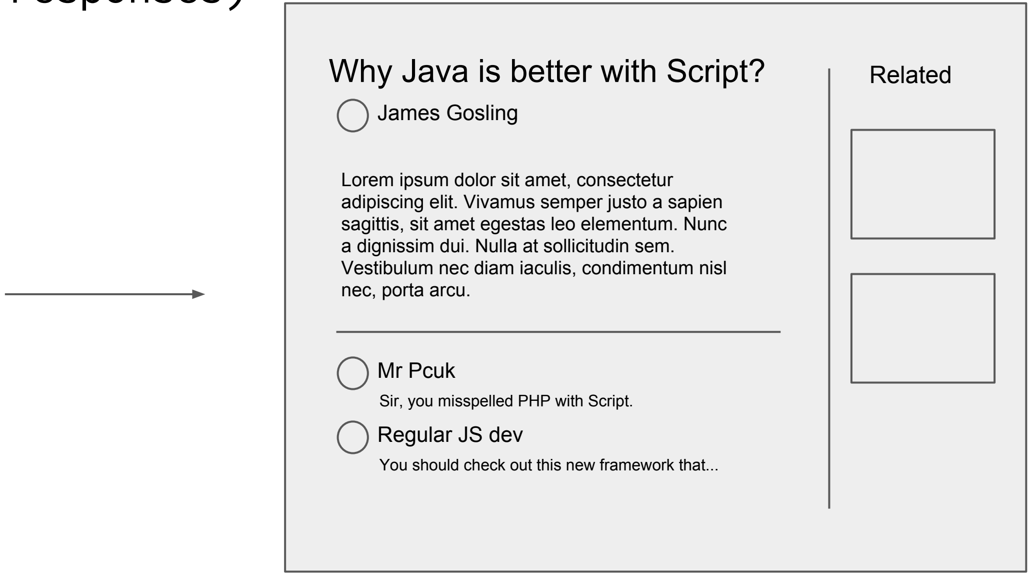
```
GET /posts/1

{

  author_id: 1,

  title: ‘Why Java is better with Script?’,

  comments: [

    { author_id: 2, content: ‘...’ },

    { author_id: 2, content: ‘...’ },

  ],

  related_posts: [1, 2, 3],

}
```


Why Java is better with Script?
James Gosling
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus semper justo a sapien sagittis, sit amet egestas leo elementum. Nunc a dignissim dui. Nulla at sollicitudin sem. Vestibulum nec diam iaculis, condimentum nisl nec, porta arcu.
Mr Pcuk
Sir, you misspelled PHP with Script.
Regular JS dev
You should check out this new framework that...
Related

Q: How many requests to API to show this page?

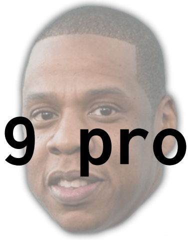
REST (in peace, waiting for responses)

```
GET /posts/1

{

  author_id: 1,

  title: ‘Why Java is better with Script?’,

  comments: [

    { author_id: 2, content: ‘...’ },

    { author_id: 2, content: ‘...’ },

  ],

  related_posts: [1, 2, 3],

}
```


Why Java is better with Script?
James Gosling
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus semper justo a sapien sagittis, sit amet egestas leo elementum. Nunc a dignissim dui. Nulla at sollicitudin sem. Vestibulum nec diam iaculis, condimentum nisl nec, porta arcu.
Mr Pcuk
Sir, you misspelled PHP with Script.
Regular JS dev
You should check out this new framework that...
Related

Q: How many requests to API to show this page?

A: about 100 000



I have 99 problems and REST is one of them

- Not optimal for mobile - multiple requests to display even a simple page part

- Combining multiple dependent data sources into single coherent view may be challenging (de/normalizing, nested structures, relations)

- To optimize bandwidth, we need granular control over what is send to the client

What if...

...**we “extend” REST?** (“aggregated views”)

```
/posts_with_author_data_and_comments_authors/
```

...we create a monster from REST?

“fields param”, “data inclusion” + mountain of hacks

```
/posts/1?include_fields=author_id,comments__first_2,related__first_3&expand=author_id__name,author_id__avatar
_url,comments__all__author_id__name
```

...we create our own DSL to solve all our problems?

What could possibly go wrong? Or: inventing GraphQL without FB resources

Enough about REST

ACT II

Scene 2: REST leaves, crying

GraphQL

GraphQL quickie

“JSON without values”

```
{
  allPosts {
    author {
      name
    }
    content
    comments {
      author { name, avatar_url }
    }
  }
}
```

REST & GraphQL - ultra quick comparison

“One core difference between REST and GraphQL — the description of a particular resource is not coupled to the way you retrieve it.” [0]

[0] https://dev-blog.apollodata.com/graphql-vs-rest-5d425123e34b

GraphQL is a **query language** for APIs and a **runtime for fulfilling** those **queries** with your existing data.

Query Language

Language

Has types:

- Scalars (primitive types) - int, float, string, boolean, ID (serializable to string), Enum

- Objects:

```
type Person {
  name: String
  age: Int
  picture: Url
}
```

...Interfaces, Unions, Lists, Fragments (“reusable objects”), null/not null constraints, btw, js is better than php, directives (@skip, @include), zmienne...

Query

Queries + Mutations + Subscription

Unexpected buzzword: CQRS!

```
mutation {
  addComment(postID: 123, text: “LOL”) {
     comment {
      id
    }
  }
}
```

Bonus: Schema Introspection

```
{
  __type(name: "User") {
    name
    fields {
      name
      type {
        name
      }
    }
  }
}
```

$\longrightarrow$

```
{
  "__type": {
    "name": "User",
    "fields": [
      {
        "name": "id",
        "type": { "name": "String" }
      },
      {
        "name": "name",
        "type": { "name": "String" }
      },
      {
        "name": "birthday",
        "type": { "name": "Date" }
      },
    ]
  }
}
```

Why? Useful for building tools

Code