

GRAPHQL

1

STOP *RESTING* ON YOUR LAURELS

GRAPHQL IS A SPEC

2

- API QUERY LANGUAGE DESIGNED FOR DATA INTENSIVE APPS
- CLIENTS DEFINE THEIR OWN DATA NEEDS
- NOT RESTRICTED TO SPECIFIC LANGUAGES OR SPECIFIC DATABASE ARCHITECTURES

GRAPHQL LIBRARIES

3

- JAVASCRIPT
- PHP
- PYTHON

- RUBY
- JAVA
- GO

GOAL OF GRAPHQL

4

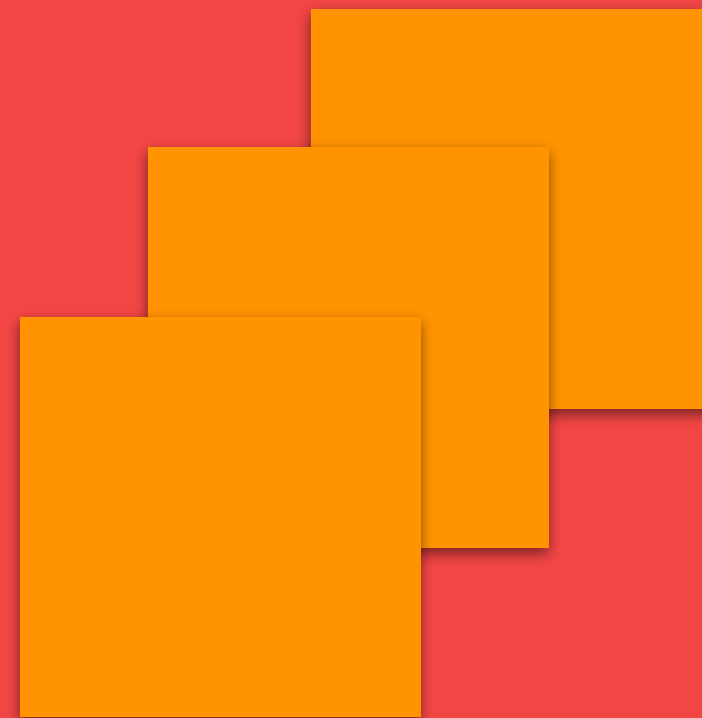
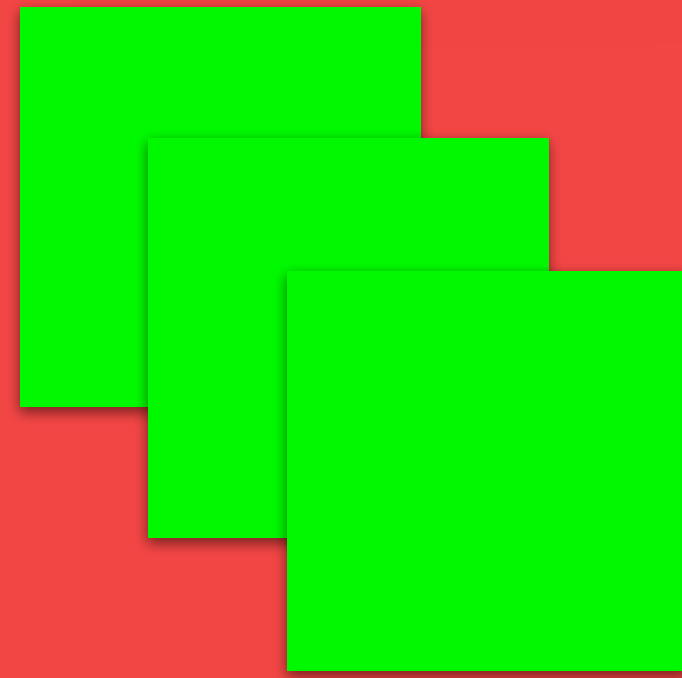
- CLIENT IS KING
- HOW DO WE MANAGE THE INCREASINGLY COMPLEX DATA REQUIREMENTS OF MODERN WEB & MOBILE APPS

DESIGN PRINCIPLES

5

- **HIERARCHICAL** → QUERIES ARE SHAPED LIKE THE DATA THEY RETURN
- **PRODUCT-CENTRIC** → DRIVEN BY THE FRONT-END REQUIREMENTS
- **STRONG-TYPING** → QUERIES ARE VALIDATED BEFORE EXECUTION
- **CLIENT-SPECIFIED QUERIES** → RETURNS ONLY WHAT A CLIENT ASKS FOR
- **INTROSPECTIVE** → PUBLISHES AND SELF-DOCUMENTS ITS CAPABILITIES

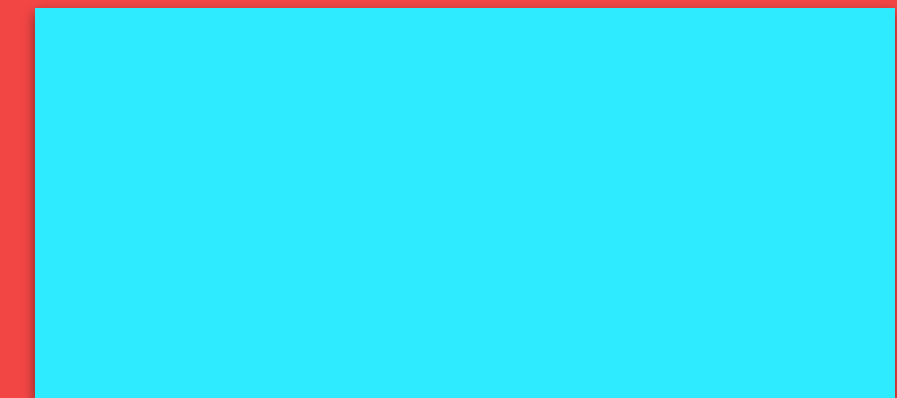
CLIENT



SERVER

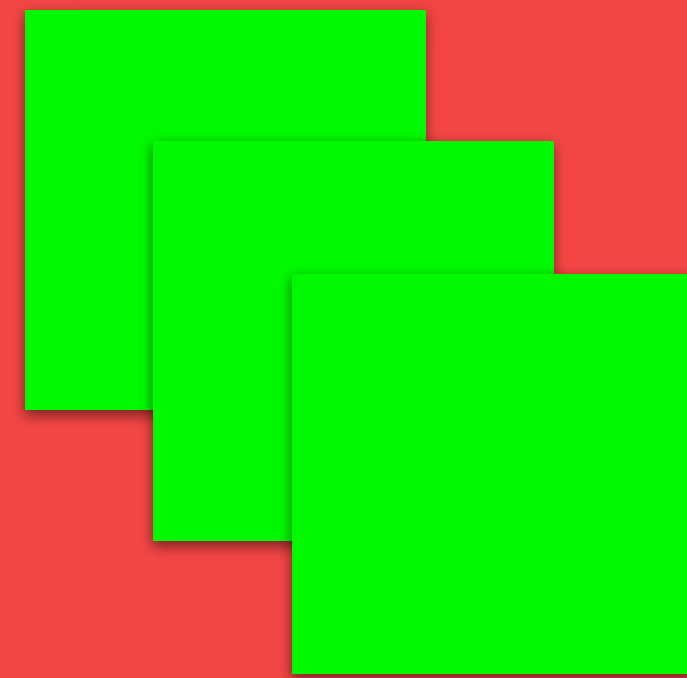


DATABASE

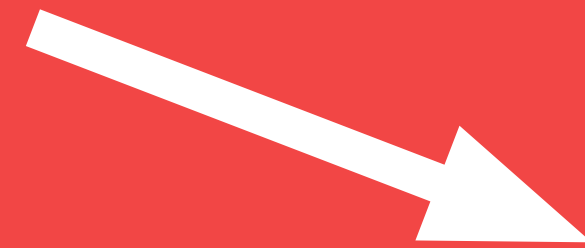


RESPONSE

CLIENT



GRAPHQL

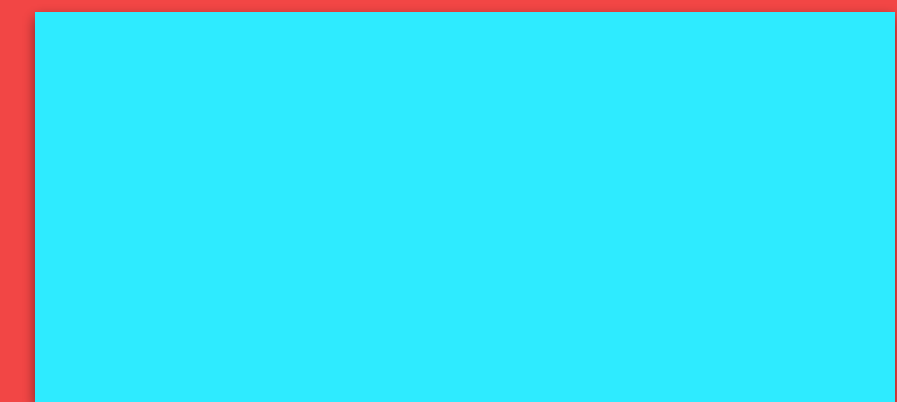
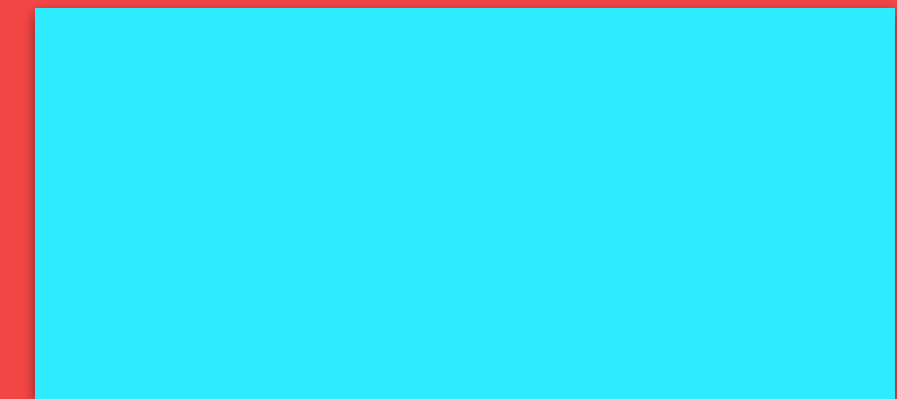


RESPONSE

SERVER



DATABASE



WHY GRAPHQL?

8

- NATURAL REPRESENTATION OF REAL DATA
- SELF-DOCUMENTING API + PLAYGROUND
- REDUCE NETWORK CALLS FOR CLIENT APPS
- DATA QUERIES ARE CATERED FOR THE CLIENT APPS
- IDEAL FOR HEAVY QUERYING AND NESTED RELATIONSHIPS

WHY NOT GRAPHQL?

9

- AWKWARD *JSON-LIKE* SYNTAX
- LOSE HTTP RESPONSE CACHING
- UNNECESSARY FOR FLAT DATA SCHEMAS
- HTTP2 & PARALLEL REQUESTS MAY DEFEAT THE PERFORMANCE GAINS
- RELAY HAS ONLY BEEN IMPLEMENTED IN REACT

HOW TO USE IT

10

- QUERYING
- CREATING & UPDATING
- BUILDING A GRAPHQL ENDPOINT

QUERYING

11

- **EXAMPLE:** FETCH THE TITLE OF RECENT BLOG POSTS AND GET EACH AUTHOR'S NAME AND TWITTER HANDLE
- COMPARING **REST**, **MAGIC BOX**, & **GRAPHQL**

QUERYING REST

12

```
GET /posts?order_by=latest
```

```
GET /authors
```

QUERYING MAGICBOX

13

```
GET /posts?include[]=authors&sort[datetime]=desc
```

QUERYING GRAPHQL

14

```
POST /graphql
query {
  posts(order_by: "desc") {
    title
  },
  authors {
    name
    twitterID
  }
}
```

CREATE & UPDATE

15

- **EXAMPLE:** CREATE A BLOG POST
- COMPARING **REST**, **MAGIC BOX**, & **GRAPHQL**

CREATE & UPDATE REST

16

```
POST /posts
{
  ...post
}
```


CREATE & UPDATE MAGICBOX

17

```
POST /posts
{
  ...post
}
```

CREATE & UPDATE GRAPHQL

18

```
POST /graphql
  mutation createPost(
    ...postInformation
  ){
    title
    author
  }
```



DEMO

RELAY & INTEGRATION

20

- SEAMLESS INTEGRATION BETWEEN **REACT & GRAPHQL**
- COMPONENTS DEFINE THEIR SPECIFIC DATA REQUIREMENTS AND API CALLS ARE ONLY MADE WHEN NEEDED

CONCLUSION

21

- GRAPHQL & REST WORK SURPRISINGLY WELL TOGETHER
- GRAPHQL ALLOWS CLIENTS FLEXIBILITY WHEN FETCHING DATA
- REST IS PERFECT FOR UPDATING AND CREATING RESOURCES

- FACEBOOK'S GRAPHQL SPEC → CANONICAL DOCUMENT
- LET'S LEARN GRAPHQL → INTRODUCTORY COURSE
- AWESOME GRAPHQL → LIST GRAPHQL PROJECTS
- GRAPHQL IS EVIL → WELL REASONED CRITICISM OF GRAPHQL
- FALCOR → BECAUSE NETFLIX HAS BRILLIANT ENGINEERS TOO
- ZERO TO GRAPHQL IN 30 MINUTES → BEST VIDEO INTRODUCTION

RESOURCES