GraphQL

Joel Corrêa
Software Engineer at @ilegra

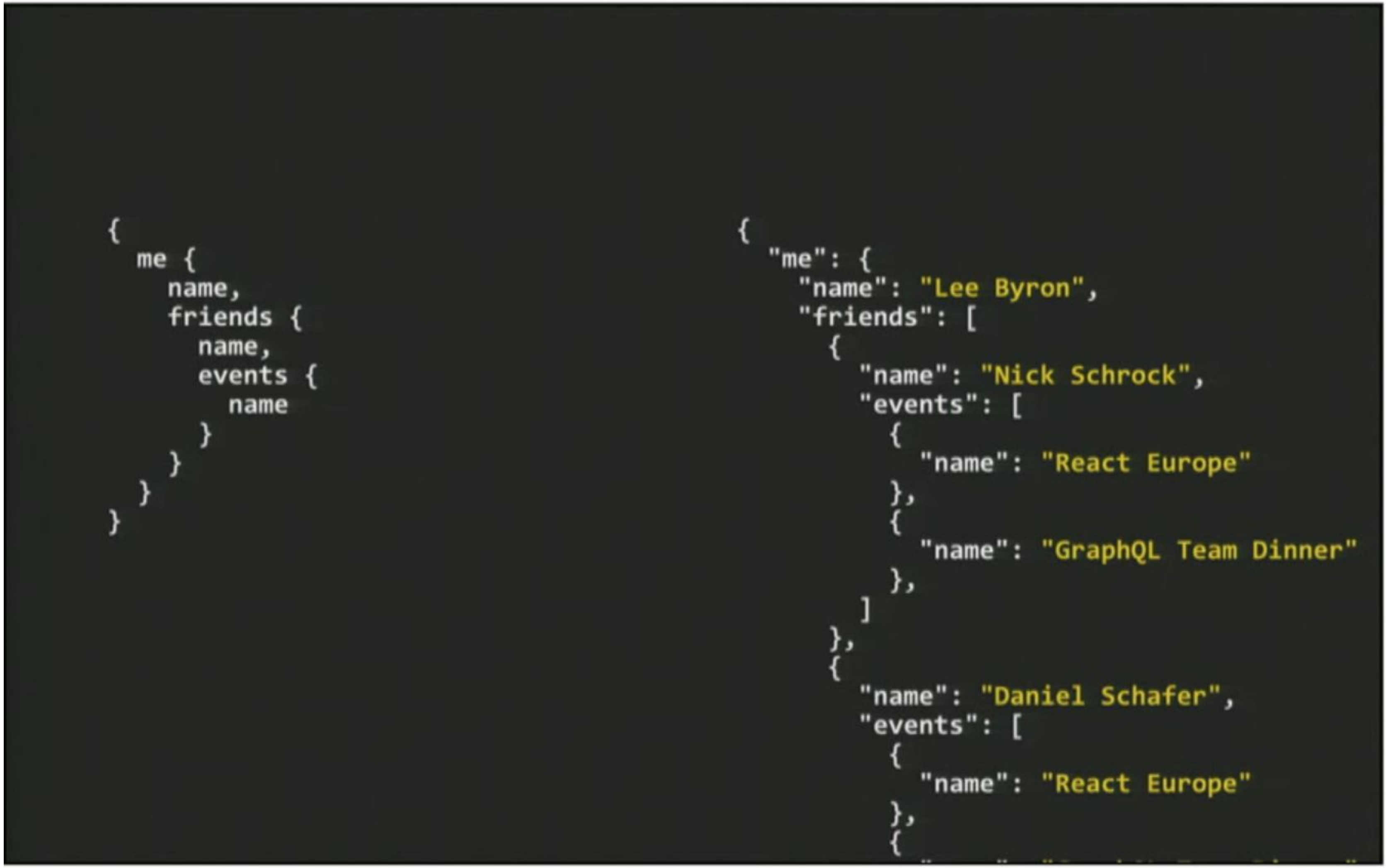
@joelcorrea_

GraphQL is a **query language** created by Facebook in 2012 which provides a common interface between the client and the server for **data fetching and manipulations**

Design considerations

Hierarchical

- ”We <u>don’t think</u> of data in terms of resource URLs, secondary keys, or join tables; we think about it in terms of a **<u>graph of objects</u>** and the models we ultimately use in our apps like NSObjects or JSON.”
- The query is shaped **<u>just like the data it returns</u>**
- Natural way for clients to describe data requirements



```
{
  me {
    name,
    friends {
      name,
      events {
        name
      }
    }
  }
}
```

```
{
  "me": {
    "name": "Lee Byron",
    "friends": [
      {
        "name": "Nick Schrock",
        "events": [
          {
            "name": "React Europe"
          },
          {
            "name": "GraphQL Team Dinner"
          },
        ]
      },
      {
        "name": "Daniel Schafer",
        "events": [
          {
            "name": "React Europe"
          },
          {
```

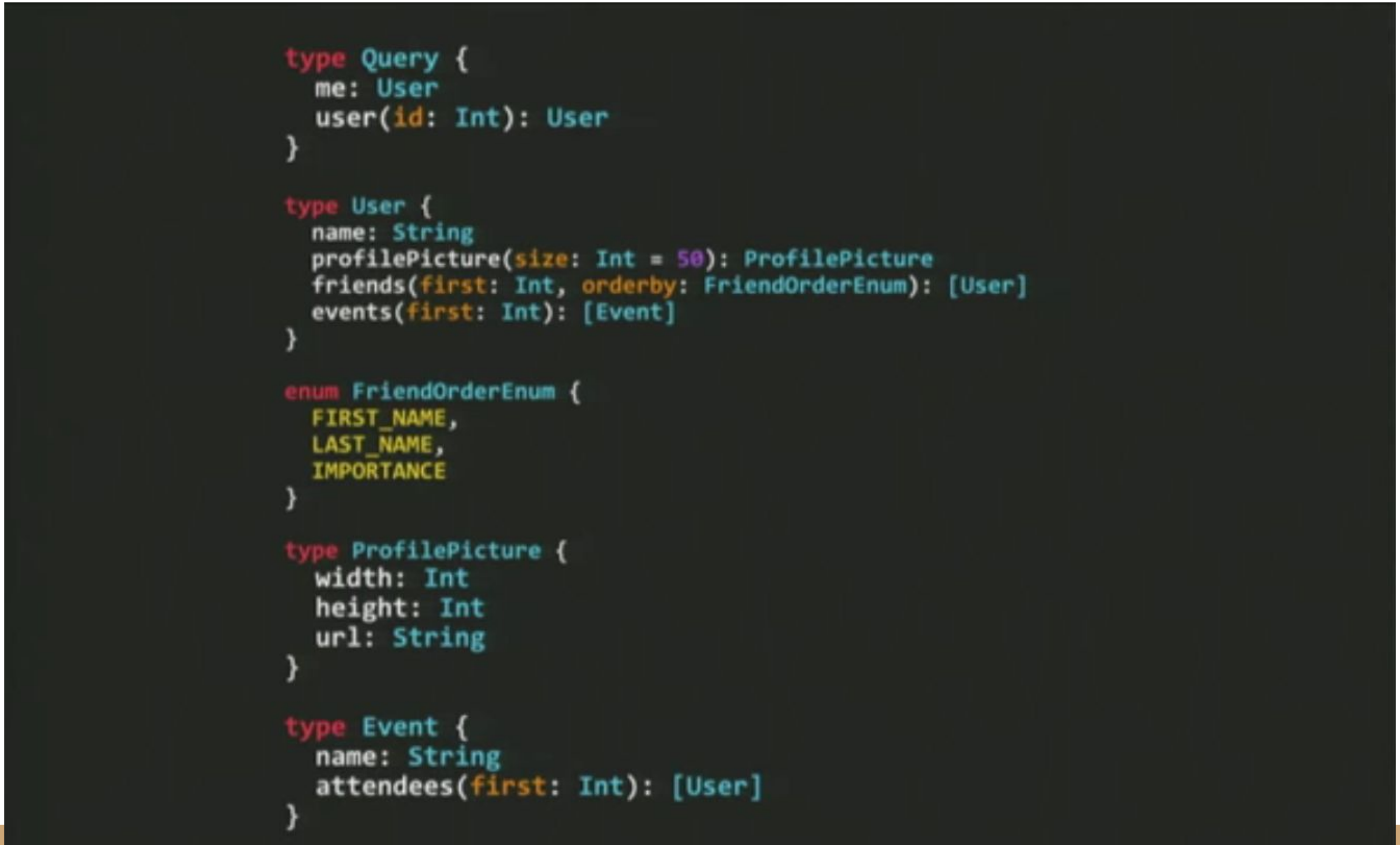
Product-centric

- Driven by the **requirements of views** and the **front-end engineers** that write them
- GraphQL starts with **their way of thinking the requirements** and build the language and runtime necessary to enable that

Strong-typing

- Every GraphQL server defines an **application-specific type system**
- Given a query, tools can ensure that the query is both syntactically correct and valid within the GraphQL type system before execution
- At the development time, and the server can make certain guarantees about the **shape and nature of the response**.

Type system



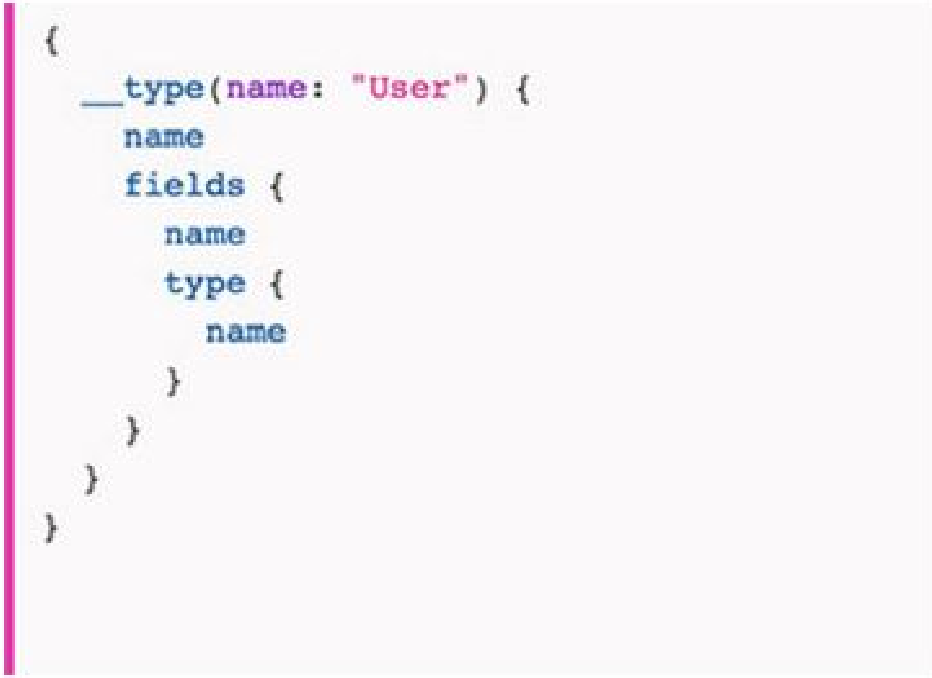
```
type Query {
  me: User
  user(id: Int): User
}

type User {
  name: String
  profilePicture(size: Int = 50): ProfilePicture
  friends(first: Int, orderby: FriendOrderEnum): [User]
  events(first: Int): [Event]
}

enum FriendOrderEnum {
  FIRST_NAME,
  LAST_NAME,
  IMPORTANCE
}

type ProfilePicture {
  width: Int
  height: Int
  url: String
}

type Event {
  name: String
  attendees(first: Int): [User]
}
```

Client-specified queries

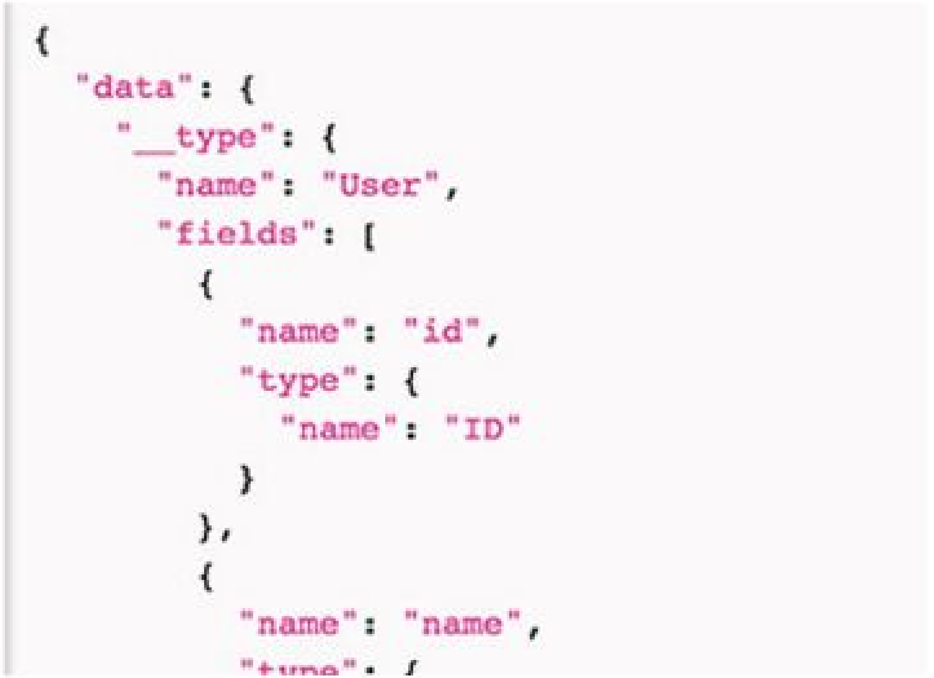
- It is the client that is responsible for specifying exactly how it will consume those published capabilities.
- These queries are specified at **field-level granularity**
- In the majority of client-server applications written without GraphQL, the server determines the data returned in its various scripted endpoints. A GraphQL query, on the other hand, returns exactly what a client asks for and no more.

Introspective

- A GraphQL server’s type system must be queryable by the GraphQL language itself



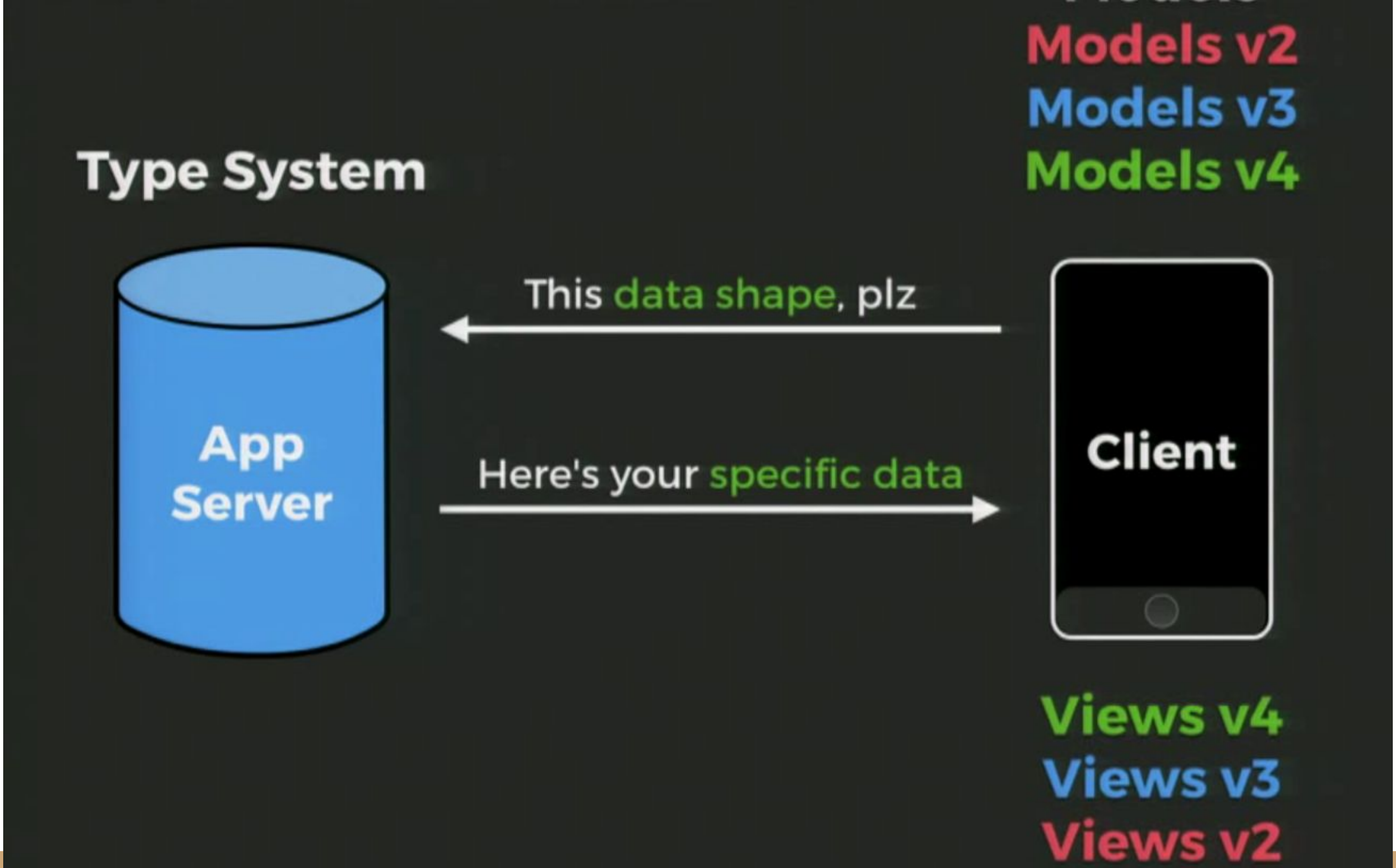
```
{
  __type(name: "User") {
    name
    fields {
      name
      type {
        name
      }
    }
  }
}
```



```
{
  "data": {
    "__type": {
      "name": "User",
      "fields": [
        {
          "name": "id",
          "type": {
            "name": "ID"
          }
        },
        {
          "name": "name",
          "type": {
```

Version free

- When you're adding **new product features**, additional fields can be added to the server, leaving existing clients unaffected. When you're **unsetting older features**, the corresponding server fields can be **deprecated** but continue to function.
- Gradual, **backward-compatible** process which removes the need for an incrementing version number
- Facebook still support **three years of released Facebook applications** on the same version of our GraphQL API


Models v2
Models v3
Models v4
Type System
This data shape, plz
App Server
Here's your specific data
Client
Views v4
Views v3
Views v2

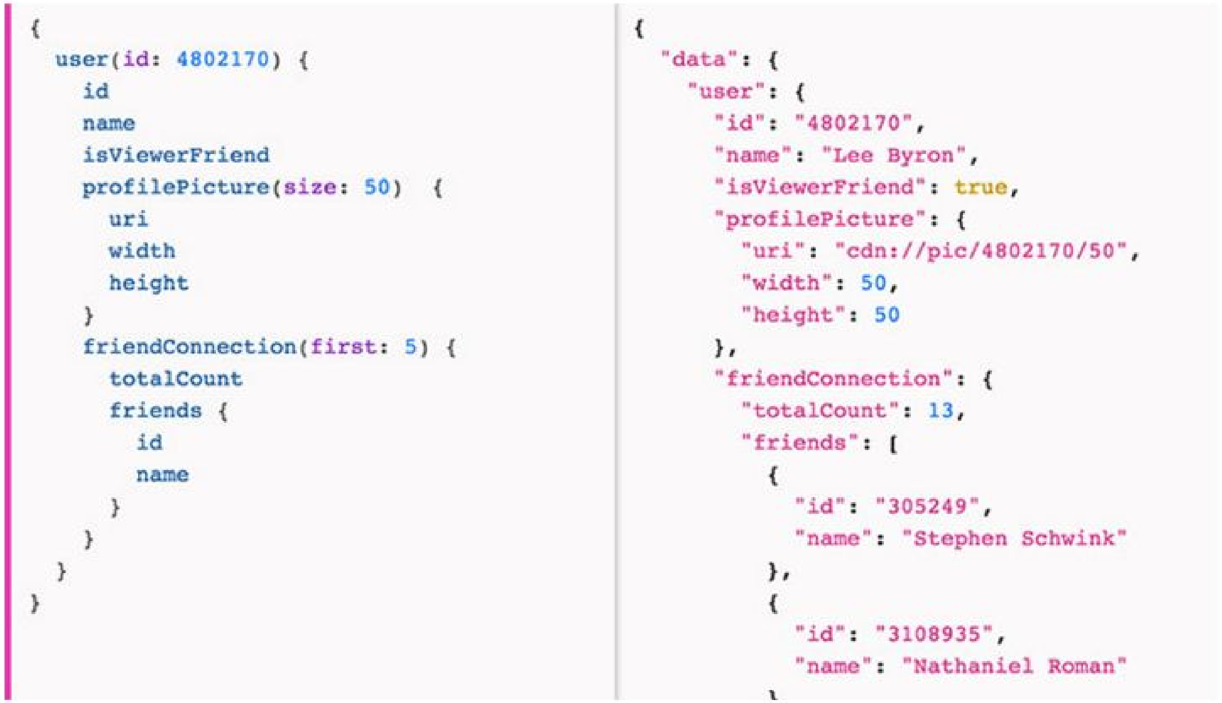
Transport independent

HTTP is just one option - GraphQL is transport independent, so you can use it with **websockets** or even **mqtt**.

Goals

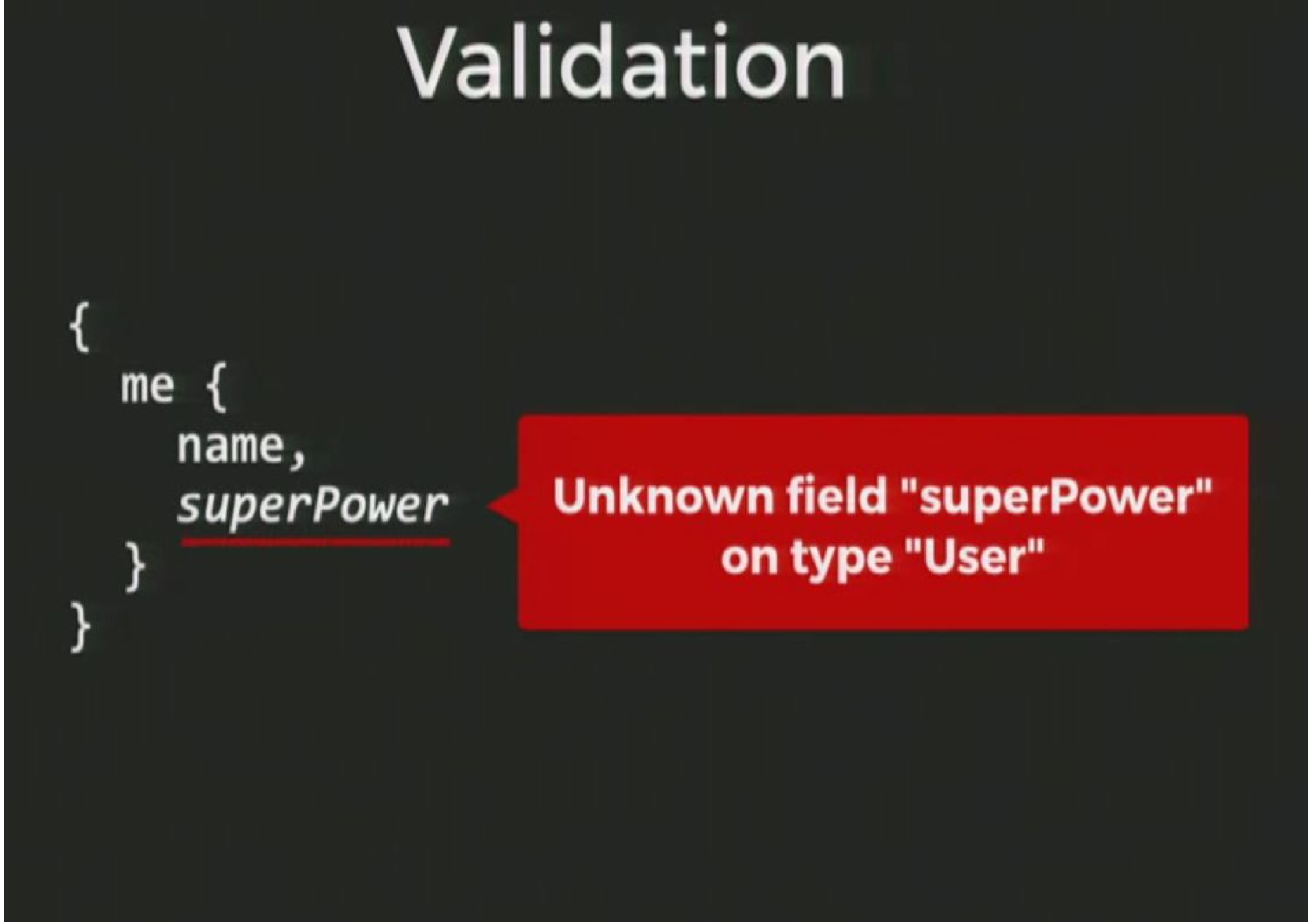
- A powerful and productive environment for building **client applications**
- **Product developers and designers** building applications against working GraphQL servers can quickly become productive without reading extensive documentation and with little or no formal training.

Query



```
{
  user(id: 4802170) {
    id
    name
    isViewerFriend
    profilePicture(size: 50)  {
      uri
      width
      height
    }
    friendConnection(first: 5) {
      totalCount
      friends {
        id
        name
      }
    }
  }
}
```

```
{
  "data": {
    "user": {
      "id": "4802170",
      "name": "Lee Byron",
      "isViewerFriend": true,
      "profilePicture": {
        "uri": "cdn://pic/4802170/50",
        "width": 50,
        "height": 50
      },
      "friendConnection": {
        "totalCount": 13,
        "friends": [
          {
            "id": "305249",
            "name": "Stephen Schwink"
          },
          {
            "id": "3108935",
            "name": "Nathaniel Roman"
```



Validation

```
{
  me {
    name,
    superPower
  }
}
```

Unknown field "superPower" on type "User"

References

- https://facebook.github.io/graphql/
- https://code.facebook.com/posts/1691455094417024/graphql-a-data-query-language/
- https://www.youtube.com/watch?v=WQLzZf34FJ8
- JS Reference Server impl:
    - https://github.com/graphql/graphql-js