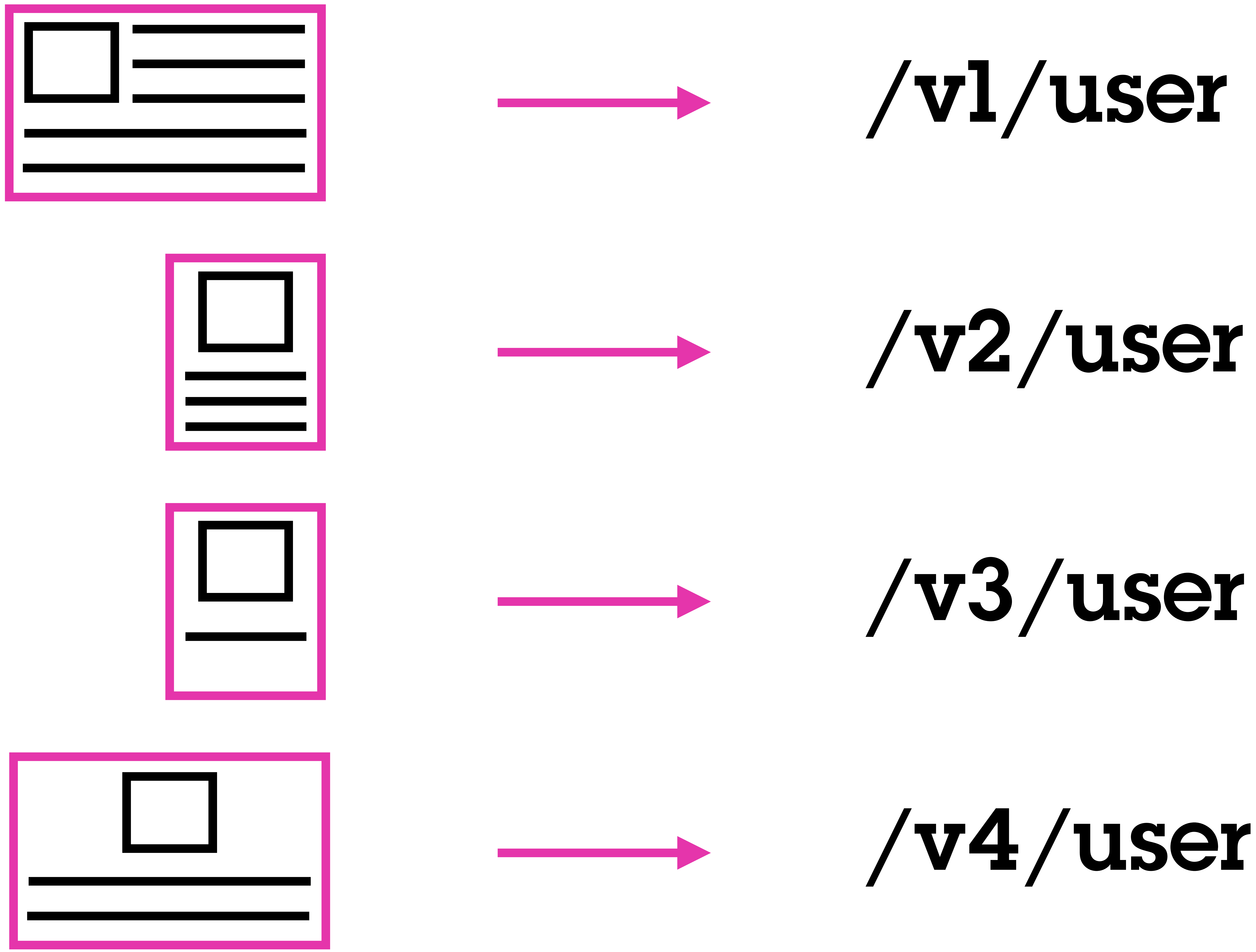
GraphQL

A DATA QUERY LANGUAGE AND RUNTIME

Cerebration Session • Jonathan Kaufman • 7/22

Motivation


/v1/user
/v2/user
/v3/user
/v4/user

Supporting
Versions
for each
App Update

Overfetching
or
Underfetching

idea: have the clients request the data they want

```
{
  user(id: 3500401) {
    id,
    name,
    isViewerFriend,
    profilePicture(size: 50)  {
      uri,
      width,
      height
    }
  }
}
```

```
{
  "user": {
    "id": 3500401,
    "name": "Chris Spencer",
    "isViewerFriend": true,
    "profilePicture": {
      "uri": "http://someurl.cdn/pic.jpg",
      "width": 50,
      "height": 50
    }
  }
}
```

```graphql
{
  user(id: 3500401) {
    name
  }
}
```

```json
{
  "user": {
    "name": "Chris Spencer"
  }
}
```

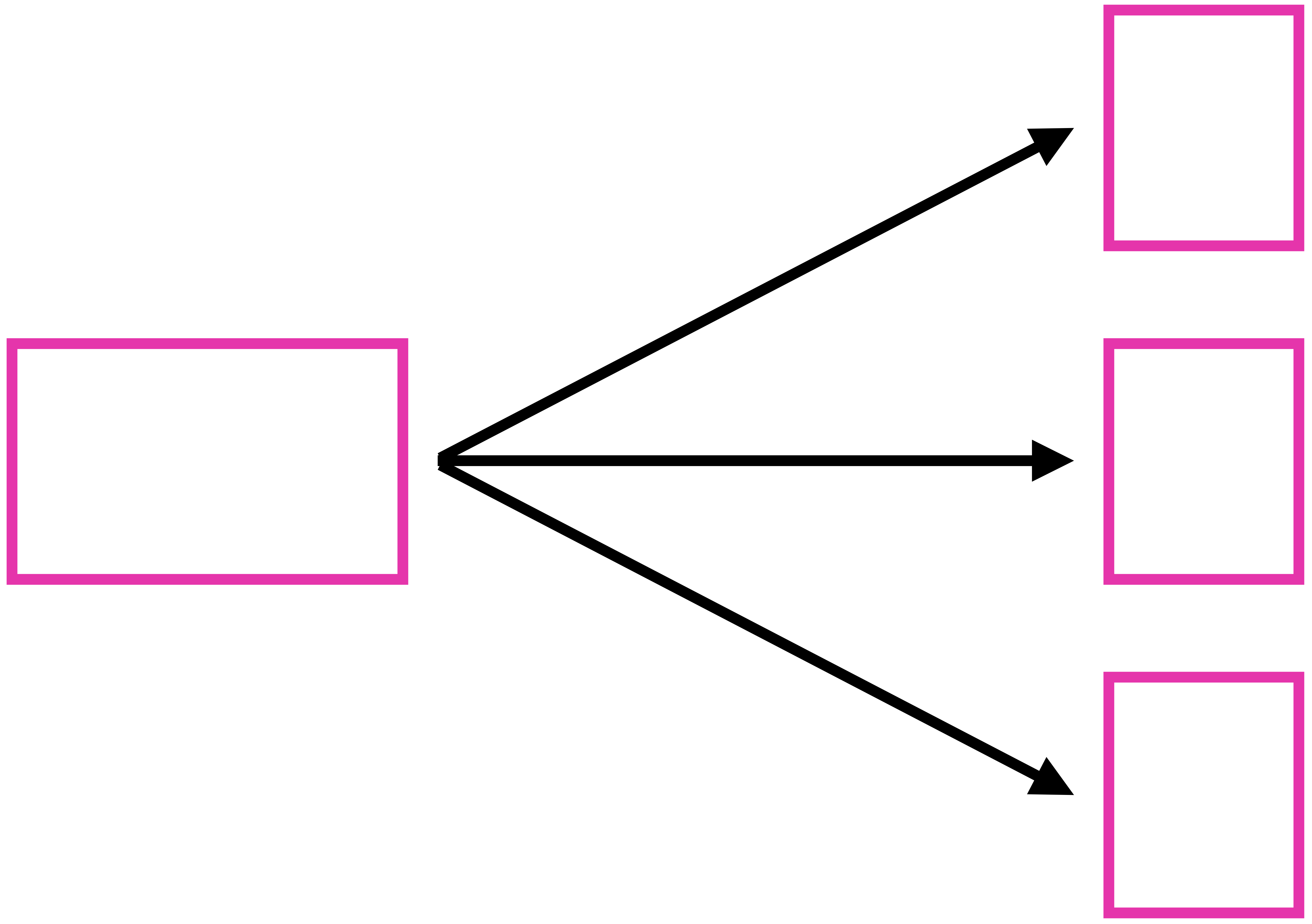
Hierarchical

The query is shaped just like the data it returns.

Client-Specific

The specification for queries are encoded in the client rather than the server. This returns exactly what a clients asks for and no more.

Avoid Round Trips



1. Get the main object which might contain an array of IDs.
2. Either perform one request for each ID or a batch request with an array of IDs.

```
{
  user(id: 3500401) {
    id
    name,
    reports {
      id,
      name
    }
  }
}
```

```
{
  "user": {
    "id": 3500401,
    "name": "Chris Spencer",
    "reports": [{
      "id": 5460124,
      "name": "Jon Kaufman"
    }, {
      "id": 5471003,
      "name": "Corey Stubbs"
    }]
  }
}
```

```graphql
{
  user(id: 3500401) {
    name,
    reports {
      name
    }
  }
}
```

```json
{
  "user": {
    "name": "Chris Spencer",
    "reports": [{
      "name": "Jon Kaufman"
    }, {
      "name": "Corey Stubbs"
    }]
  }
}
```

Composable

Create query fragments and share them between objects.

```
{
  user(id: 3500401) {
    ...UserInfo
    reports {
      ...UserInfo
    }
  }
}

fragment UserInfo {
  id,
  name,
  profilePicture(size: 50)  {
    uri
  }
}
```

```
{
  "user": {
    "id": 3500401,
    "name": "Chris Spencer",
    "profilePicture": {
      "uri": "http://someurl.cdn/pic1.jpg"
    },
    "reports": [{
      "id": 5460124,
      "name": "Jon Kaufman",
      "profilePicture": {
        "uri": "http://someurl.cdn/pic2.jpg"
      }
    }, {
      "id": 5471003,
      "name": "Corey Stubbs",
      "profilePicture": {
        "uri": "http://someurl.cdn/pic3.jpg"
      }
    }]
  }
}
```

Building a Schema

```
const userType = new graphql.GraphQLObjectType({
  name: 'User',
  fields: {
    id: { type: graphql.GraphQLString },
    name: { type: graphql.GraphQLString },
  }
});
```

```
const schema = new graphql.GraphQLSchema({
  query: new graphql.GraphQLObjectType({
    name: 'Query',
    fields: {
      user: {
        type: userType,
        args: {
          id: { type: graphql.GraphQLString }
        },
        resolve: (_, args) => goOffToDB(args.id);
      }
    }
  })
});
```

Strongly Typed

A GraphQL query can be ensured to be valid within a GraphQL type system at development time allowing the server to make guarantees about the response.

GraphQLEnumType
GraphQLInterfaceType
GraphQLObjectType
GraphQLList
GraphQLNonNull
GraphQLSchema
GraphQLString

Piecing it all Together

```javascript
const graphql = require('graphql');
const graphqlHTTP = require('express-graphql');
const express = require('express');

...

express()
  .use('/graphql', graphqlHTTP({ schema: schema, pretty: true }))
  .listen(3000);
```

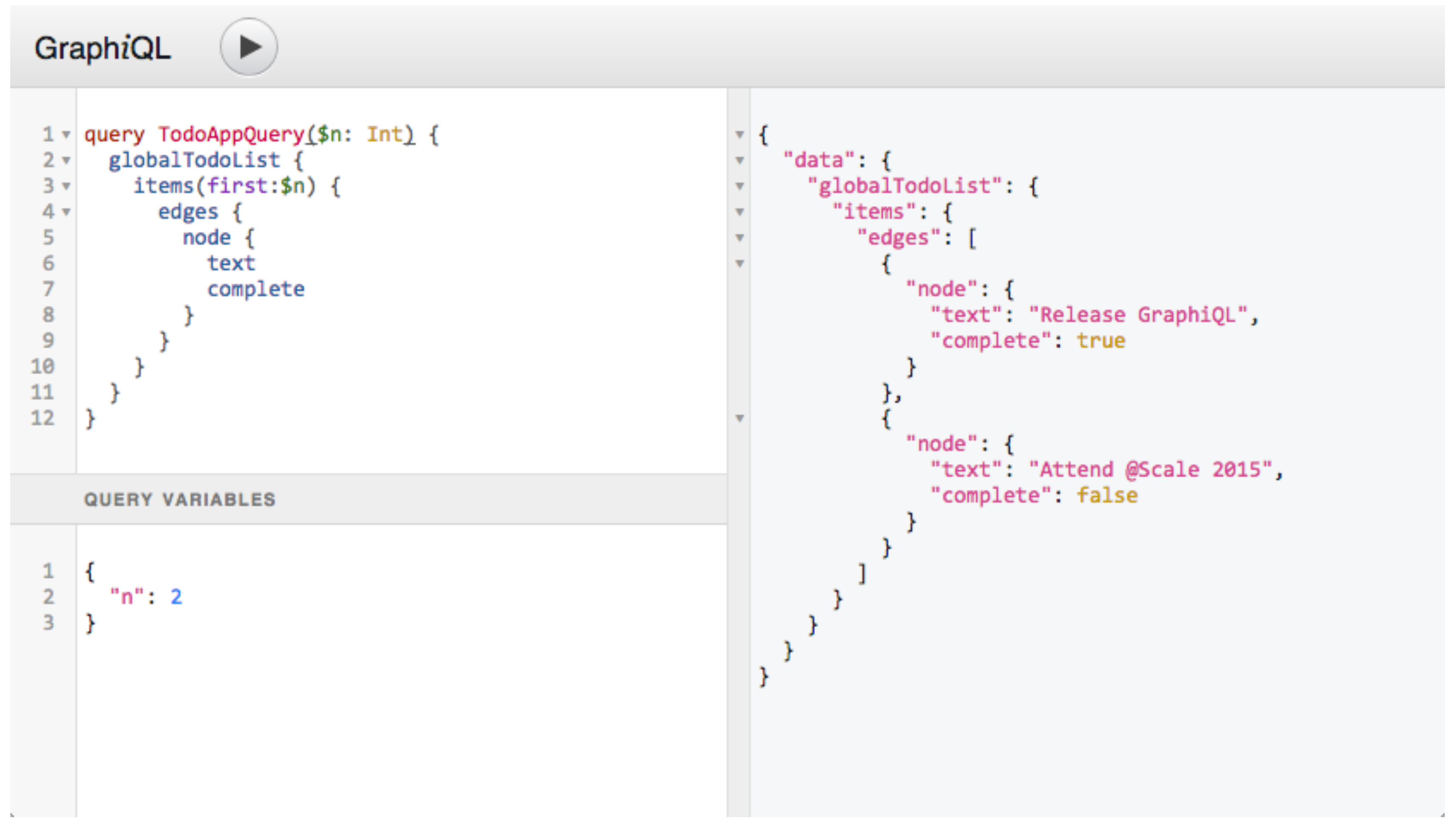
Not Covered…

Mutations
Optimistic Updates
Deletions
Introspection
Caching
Rolling up Queries

Community

Implementations

JS	Scala
PHP	.NET
Ruby	Elixir
Python	Haskell
C/C++	SQL
Go	Lua



```
query TodoAppQuery($n: Int) {
  globalTodoList {
    items(first:$n) {
      edges {
        node {
          text
          complete
        }
      }
    }
  }
}
```

```
{
  "n": 2
}
```

```
{
  "data": {
    "globalTodoList": {
      "items": {
        "edges": [
          {
            "node": {
              "text": "Release GraphiQL",
              "complete": true
            }
          },
          {
            "node": {
              "text": "Attend @Scale 2015",
              "complete": false
            }
          }
        ]
      }
    }
  }
}
```

Client Side

Adrenaline
Apollo
Relay

https://github.com/chentsulin/awesome-graphql more info at…

http://graphql.org/