

Choosing **GraphQL** for your **API**

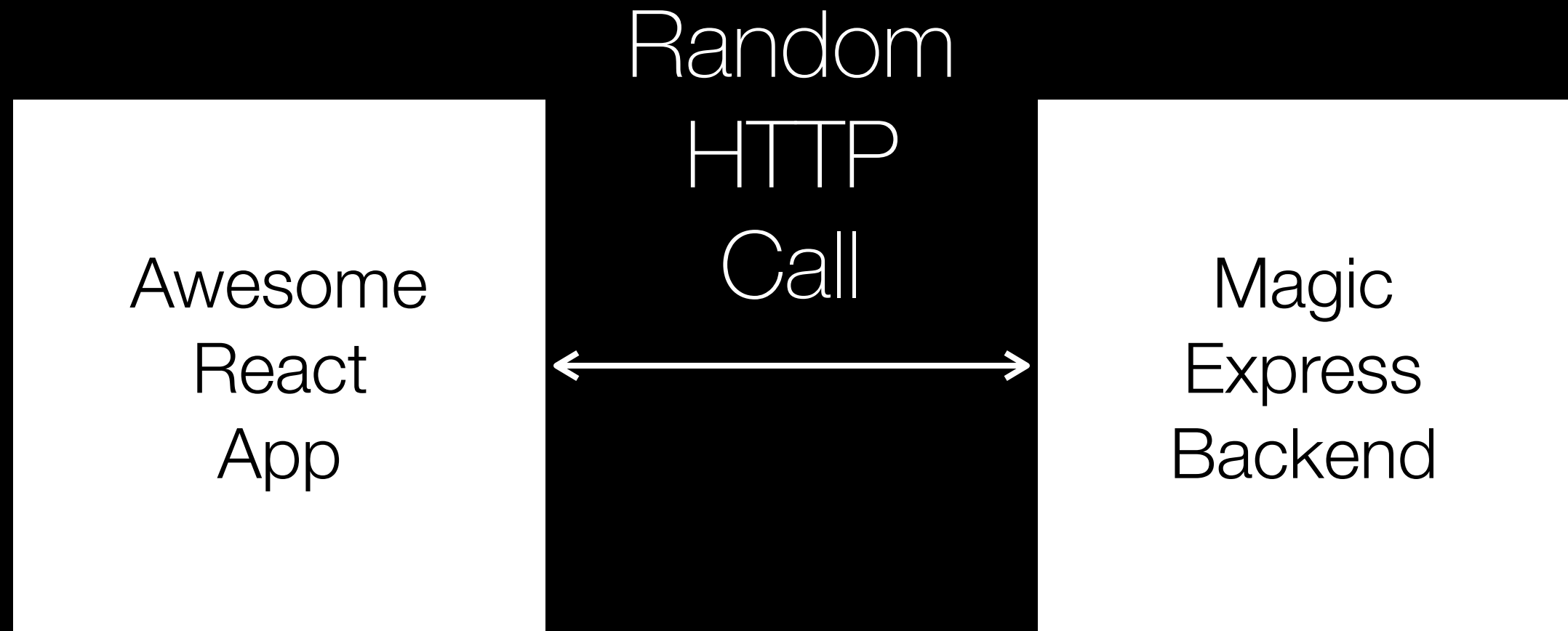
@WesleydeSouza
Developer, Work & Co

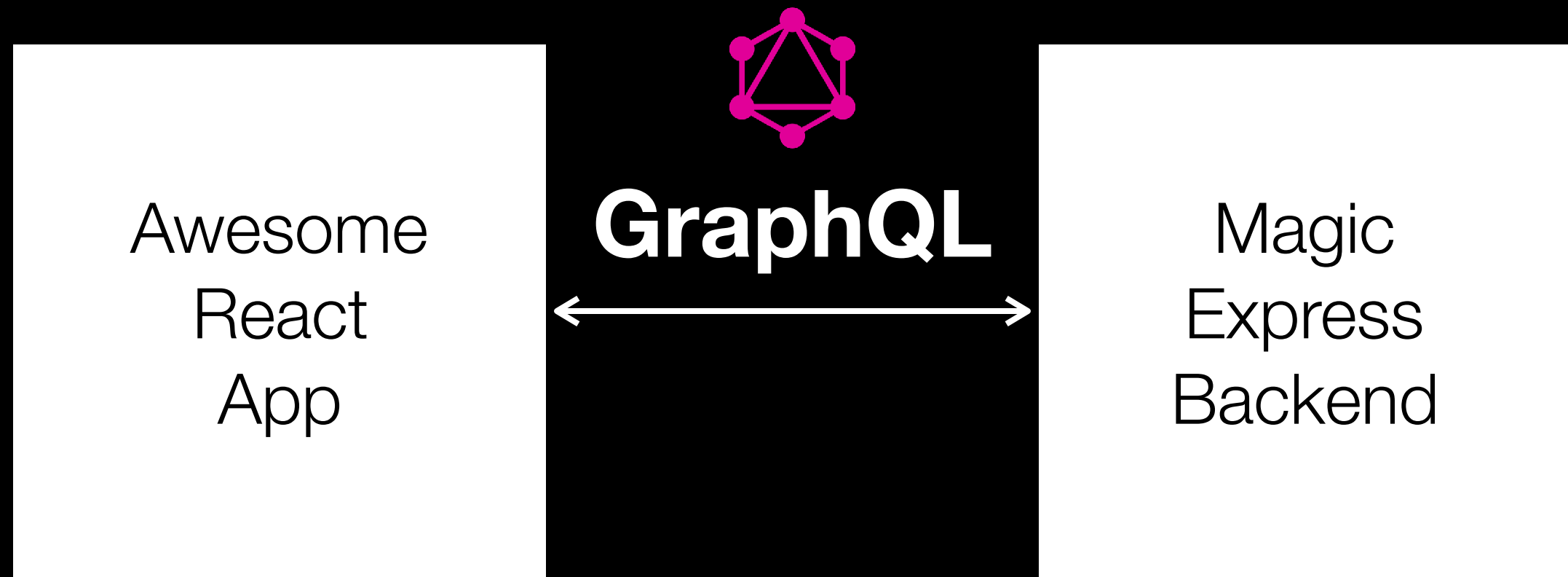


What is **GraphQL**?

A query language for your **API**

—graphql.org





Why GraphQL?

- A strong **API contract**
- Results are **predictable**, and the **client decides** what to receive
- Several **tools**, **libraries** and **services** are already available
- **Many projects** use it today

An **example**

Current Meetup Page

Awesome Meetups Website

BrooklynJS

BrooklynJS is a monthly meeting of JavaScript developers which happens on the third Thursday of each month in the upstairs event space at 61 Local, a restaurant and bar in Cobble Hill, Brooklyn.

\$15 Sold Out

Current Wishlist Page

Awesome Meetups Website

My Wishlist

[BrooklynJS \(\\$15\)](#)

[Rick and Morty Enthusiasts \(Free\)](#)

New feature

New Navigation Component

Awesome Meetups Website

Wishlist

BrooklynJS

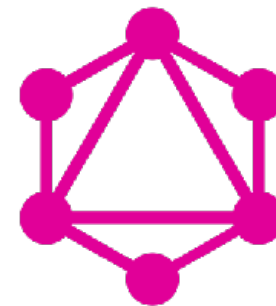
BrooklynJS is a monthly meeting of JavaScript developers which happens on the third Thursday of each month in the upstairs event space at 61 Local, a restaurant and bar in Cobble Hill, Brooklyn.

~~\$15~~ Sold Out

[BrooklynJS](#)

[Rick and Morty En...](#)

npm install



The **contract**

Schema

```
type Meetup {  
  id: ID!  
  name: String!  
  description: String!  
  price: Int!  
  priceFormatted: String!  
  isAvailable: Boolean!  
  # Dozens of other properties  
}
```


Schema

```
type Meetup {  
  id: ID!  
  # You remember the rest...  
}  
  
type Wishlist {  
  meetups: [Meetup!]!  
}
```

Schema

```
type Meetup {  
  id: ID!  
  # You remember the rest...  
}  
  
type Wishlist {  
  meetups: [Meetup!]!  
}  
  
type RootQuery {  
  meetup(id: ID!): Meetup  
  wishlist: Wishlist  
}
```

Schema

```
type Meetup {  
  id: ID!  
  # You remember the rest...  
}
```

```
type Wishlist {  
  meetups: [Meetup!]!  
}
```

```
type RootQuery {  
  meetup(id: ID!): Meetup  
  wishlist: Wishlist  
}
```

```
schema {  
  query: RootQuery  
}
```

The **data**

POST /graphql (request)

```
query GetMeetup($id: ID!) {  
  meetup(id: $id) {  
    id  
    name  
    priceFormatted  
    isAvailable  
  }  
}
```

POST /graphql (response)

```
{  
  "id": "1",  
  "name": "BrooklynJS Ticket",  
  "priceFormatted": "$15",  
  "isAvailable": false  
}
```


POST /graphql (request)


```
query GetMyWishlist {  
  wishlist() {  
    meetups {  
      id  
      name  
    }  
  }  
}
```

POST /graphql (response)


```
{  
  "meetups": [  
    {  
      "id": "1",  
      "name": "BrooklynJS Ticket"  
    },  
    // ...  
  ]  
}
```


Let's **build it**

1. Build a visual wishlist on the header
2. Write the GraphQL query
3. Connect the component to the query
4. 

1. ~~Build a visual wishlist on the header~~
2. Write the GraphQL query
3. Connect the component to the query
4. 

⌘ + Tab

1. ~~Build a visual wishlist on the header~~
2. ~~Write the GraphQL query~~
3. ~~Connect the component to the query~~
4. 

GraphQL **is great**

GraphQL **isn't perfect**

- There is no support for binary or file uploads
- GraphQL is not a database
- Optimizing the underlying system queries is your responsibility
- API endpoint can't leverage the browser's cache

Thank you

[github.com/](#)

[twitter.com/ WesleydeSouza](#)

[instagram.com/](#)

[wesley.so](#)

[graphql.org](#)

[dev.apolldata.com](#)