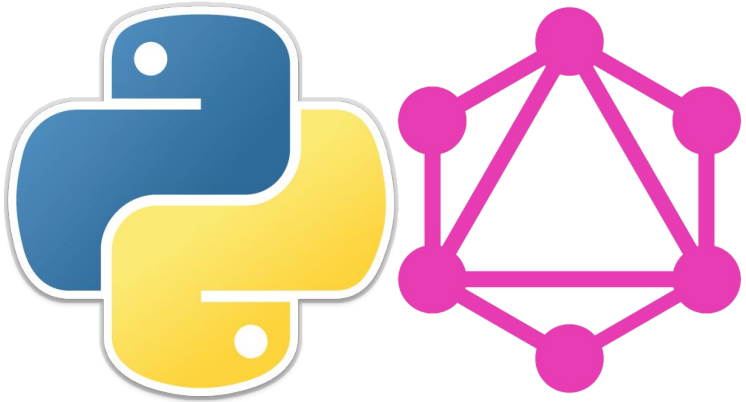
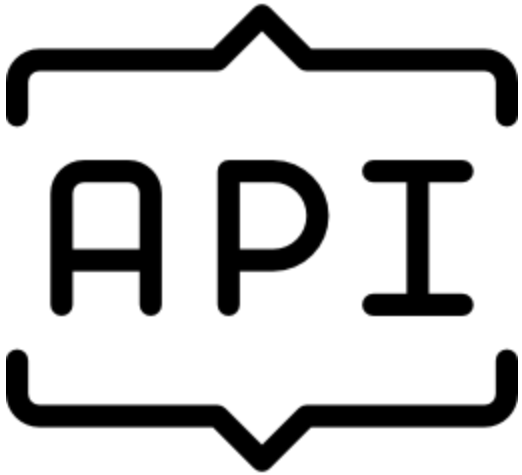


Python and GraphQL

Alec MacQueen
Software Engineer @ Administrate


API

vs

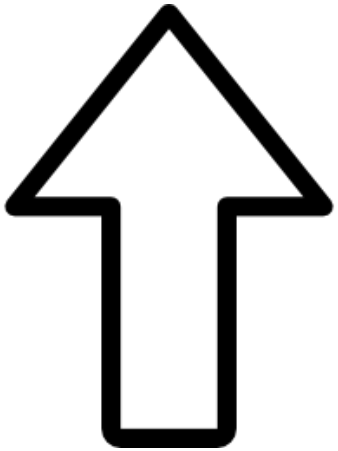
- Maintainability
- Coupling of frontend and backend
- Cognitive load
- Not enough thought given to our API

What is GraphQL?

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

graphql.org

Not language specific



A single endpoint

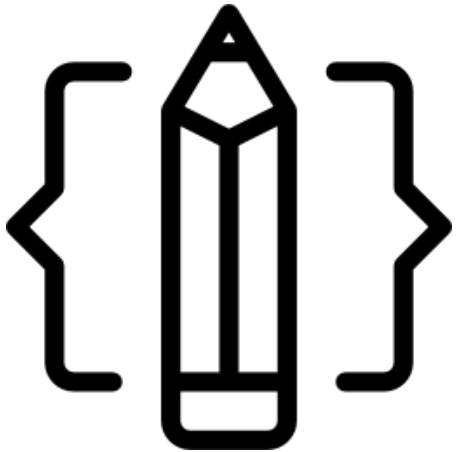
Get what you want

```
query {
  event(eventId:"1") {
    title
    description
    talks {
      name
    }
  }
}
```

```
{
  "data": {
    "event": {
      "title": "Europython 2018",
      "description": "The 2018 European
Python conference held in Edinburgh",
      "talks": [
        {
          "name": "Python and GraphQL"
        },
        {
          "name": "Hi my name is README"
        }
      ]
    }
  }
}
```

/api/events/1

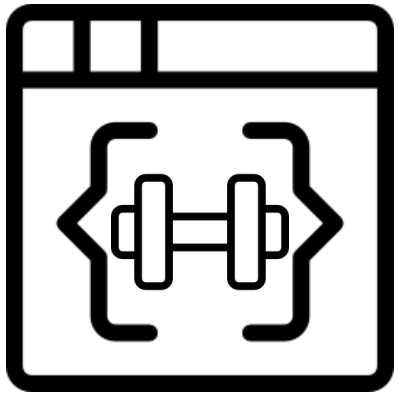
```
{
  "prices": [
    {
      "price_level": {
        "expiry_offset": null,
        "description": null,
        "is_deleted": false,
        "id": 1,
        "name": "Normal"
      },
      "event_id": 1,
      "region": {
        "name": "Vancouver",
        "default_tax": null,
        "countries": [

        ],
        "invoice_numbering": "region",
```



Mutations

```
mutation {
  account {
    create(input: {
      name: "New Account",
      emailAddress: "test@test.co"
    }) {
      account {
        name
        emailAddress
      }
    }
  }
}
```

```
{
  "data": {
    "account": {
      "create": {
        "account": {
          "name": "New Account",
          "emailAddress": "test@test.co"
        }
      }
    }
  }
}
```



Strongly typed

Let’s build an API

GRAPHQL VOYAGER

github.com/APIs-guru/graphql-voyager

GraphQL-Core

github.com/graphql-python/graphql-core

```
event_type = GraphQLObjectType(
    name='event',
    fields={
        'name': GraphQLField(
            type=GraphQLString,
            resolver=lambda root, args, *_: "Europython 2018"
        ),
        'description': GraphQLField(
            type=GraphQLString,
            resolver=lambda root, args, *_: "The best event of the year, amirite?"
        ),
        'talks': GraphQLField(
            type=GraphQLList(GraphQLString),
            resolver=lambda root, args, *_: ["Python and GraphQL"]
        )
    }
)
```

```
schema = GraphQLSchema(
  query=GraphQLObjectType(
    name='RootQueryType',
    fields={
        'event': GraphQLField(
            event_type,
            resolver=lambda root, args, *_: event_type
        )
    }
  )
)
```

```
app = Flask(__name__)
app.add_url_rule(
    '/graphql',
    view_func=GraphQLView.as_view(
        'graphql',
        schema=schema,
        graphiql=True
    )
)
```

github.com/graphql-python/flask-graphql

Graphene

github.com/graphql-python/graphene

```python
class Event(graphene.ObjectType):
    id = graphene.ID()
    name = graphene.String()
    description = graphene.String()
    talks = graphene.List(of_type=Talk)

    def resolve_talks(self, info):
        return [talks_data.get(talk_id) for talk_id in self.talks]


class Talk(graphene.ObjectType):
    name = graphene.String()
```

```
class Query(graphene.ObjectType):
    event = graphene.Field(Event, event_id=graphene.String())
    talks = graphene.Field(graphene.List(of_type=Talk))

    def resolve_event(self, info, event_id):
        return events_data.get(event_id)

    def resolve_talks(self, info):
        return [talk for talk in talks_data.values()]

schema = graphene.Schema(query=Query)
```

Graphqlize

```python
class Event:

    def __init__(self, event: dict):
        self.event = event

    def name(self) -> str:
        return self.event.get('name')

    def description(self) -> str:
        return self.event.get('description')

    def talks(self) -> List[Talk]:
        return get_talks(self.event.get('talks'))
```

```python
class Talk:

    def __init__(self, name: str):
        self.name = name

    def name(self) -> str:
        return self.name
```

```
class QueryType:

    @staticmethod
    def event(event_id: str) -> Event:
        return get_event(event_id)

    @staticmethod
    def talks() -> List[Talk]:
        return get_all_talks()


registry = graphqlize.Registry()
graphql_query_type = registry.object_type(QueryType)
schema = GraphQLSchema(graphql_query_type)
```

Can I use it? Not yet.

bit.ly/graphqlize

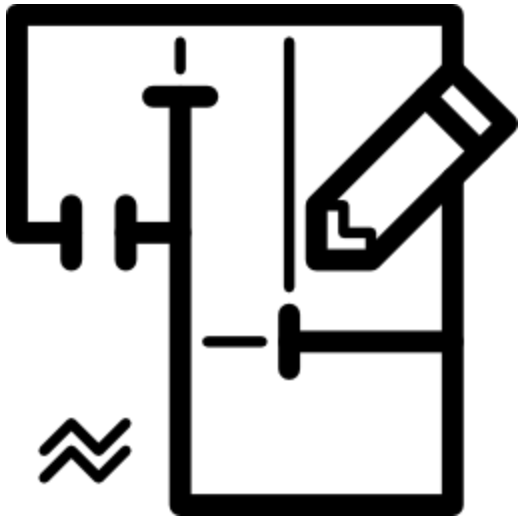
inspect

typing

Graph*i*QL

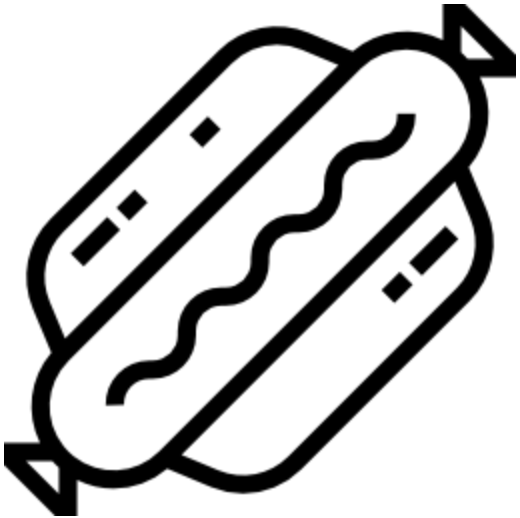
github.com/graphql/graphiql

API First



Upfront Design

Release publicly, regularly



Dogfooding

Documentation

- Maintainability
- Coupling of frontend and backend
- Cognitive load
- Not enough thought given to our API

Thank you

@macqueenism

All artwork used in this talk was designed by Vitaly Gorbachev from flaticon.com