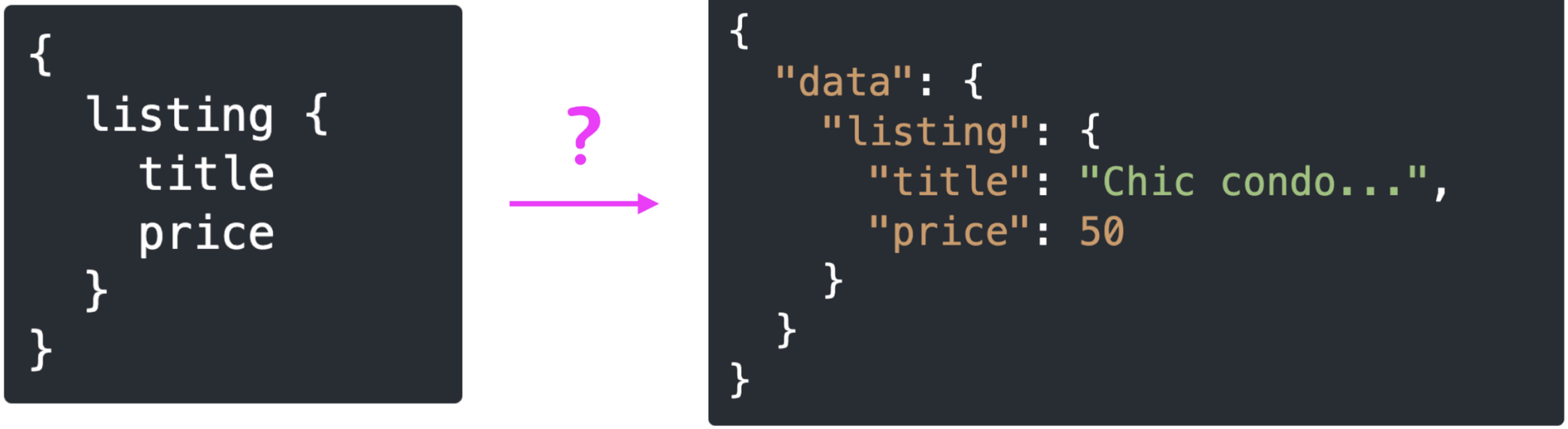
MODULE 2
2.3 - GRAPHQL CONCEPTS


{
listing {
title
price
}
}
?
{
"data": {
"listing": {
"title": "Chic condo...",
"price": 50
}
}
}

A GraphQL Schema *describes* the GraphQL API

Two large pieces of a GraphQL API

- The Schema
- The Schema Resolvers

Object Type - Listing

```
type Listing {
  id: ID!
  title: String!
  address: String!
  price: Int!
}
```

Object Type - Listing

```
type Listing {
  "..."
  tenant: User!
}
```

Object Type - User

```
type User {
  "..."
  listing: Listing!
}
```

Object Type - User

```
type User {
  "..."
  listings: [Listing!]!
}
```

Query and Mutation Object Types

```
type Query {
  listings: [Listing!]!
}
```

```
type Mutation {
  deleteListing(id: ID!): Listing!
}
```

Scalar Types

```
field: Boolean!
field: Int!
field: Float!
field: String!
field: ID!
```

Enum Types

```
enum ListingType {
  HOUSE
  APARTMENT
}
```

List Types

```
type Listing {
  "..."
  bookings: [Booking]
}
```

Arguments

```
type Mutation {
  createListing(
    id: ID!
    title: String!
    address: String!
    price: Int!
  ): Listing!
}
```

Arguments & Input Types

```
input CreateListingInput {
  id: ID!
  title: String!
  address: String!
  price: Int!
}
```

```
type Mutation {
  createListing(input: CreateListingInput!): Listing!
}
```

How do these fields resolve to return data?

Resolvers are functions that *resolve* and generate a response for a field

Root fields & Resolvers

```
Query: {
  listings: () => {
    return listings;
  },
}
```

Root fields & Resolvers

```
Mutation: {
  deleteListing: (obj, args, ctx) => {
    for (let i = 0; i < listings.length; i++) {
      if (listings[i].id === args.id) {
        return listings.splice(i, 1)[0];
      }
    }
  },
}
```

Field Resolvers

```
Listing: {
  id: (obj) => obj.id,
  title: (obj) => obj.title,
  "..."
}
```

Resolver Arguments

1. **obj** - object returned from the *parent* resolver
2. **args** - *arguments* provided to the field
3. **context** - value provided to *every* resolver
4. **info** - info about the *execution* state of the query