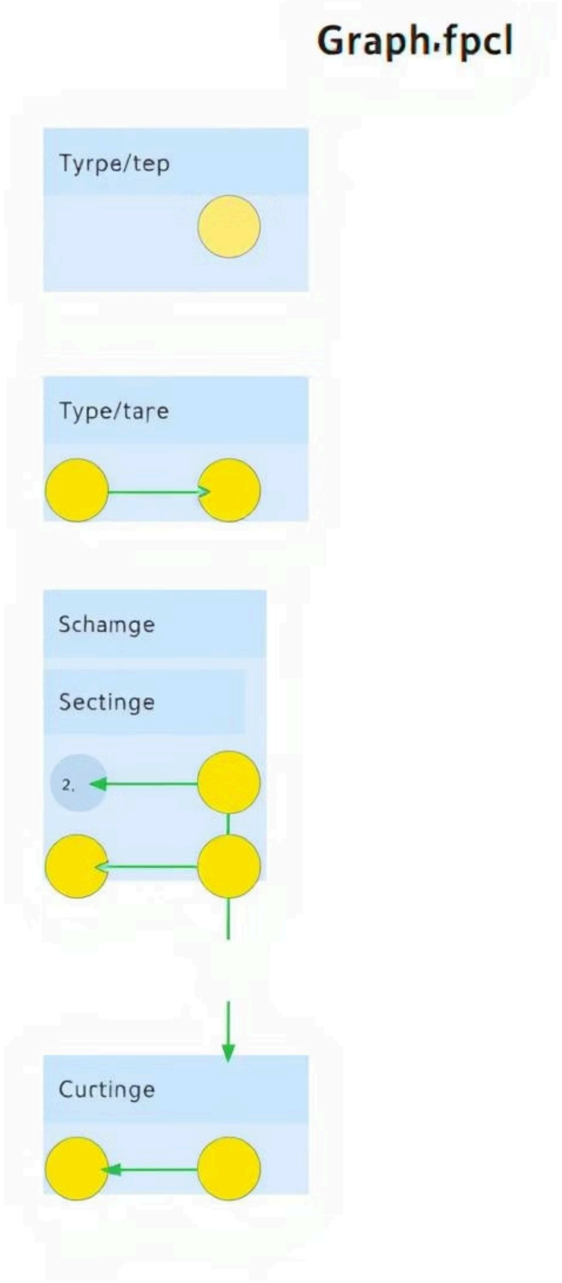
GraphQL: A Beginner's Guide

GraphQL is a modern query language for APIs. It helps clients request exactly the data they need.

This presentation will introduce GraphQL basics for new developers.

by Gautam mukherjee



Core Concepts: Schemas & Types

Schema

Defines data structure as a contract between client and server.

Types

- Scalar: Int, String, Boolean
- Objects, Lists, Non-Null

Example

type User { id: ID!, name: String, email: String }

Queries: Asking for Specific Data

Purpose

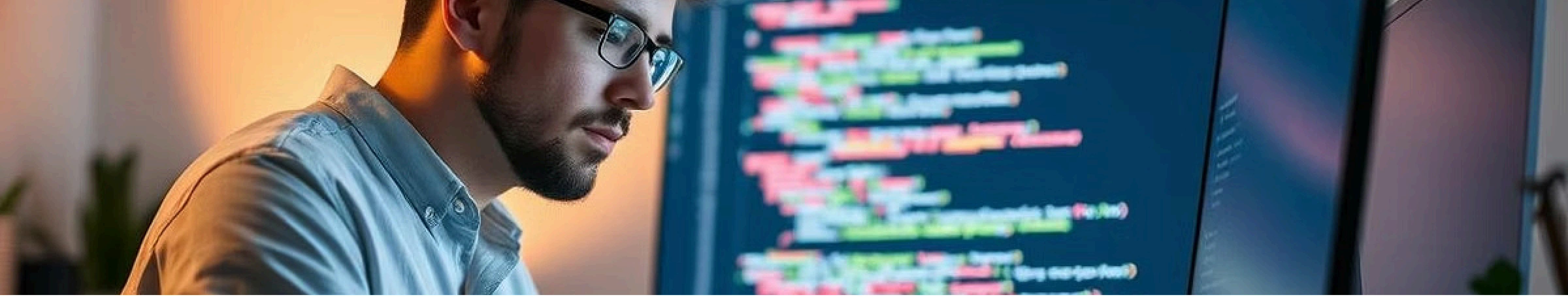
Request only the data fields you need from the server.

Example

{ user(id: "123") { name, email } }

Benefits

- Avoids over-fetching
- Prevents under-fetching



Mutations: Modifying Data

1

Create

Add new data entries.

2

Update

Modify existing data.

3

Delete

Remove data securely.

Example: mutation { createUser(name: "John", email: "john@example.com") { id name email } }

Resolvers: Connecting Schema to Data

Function Role

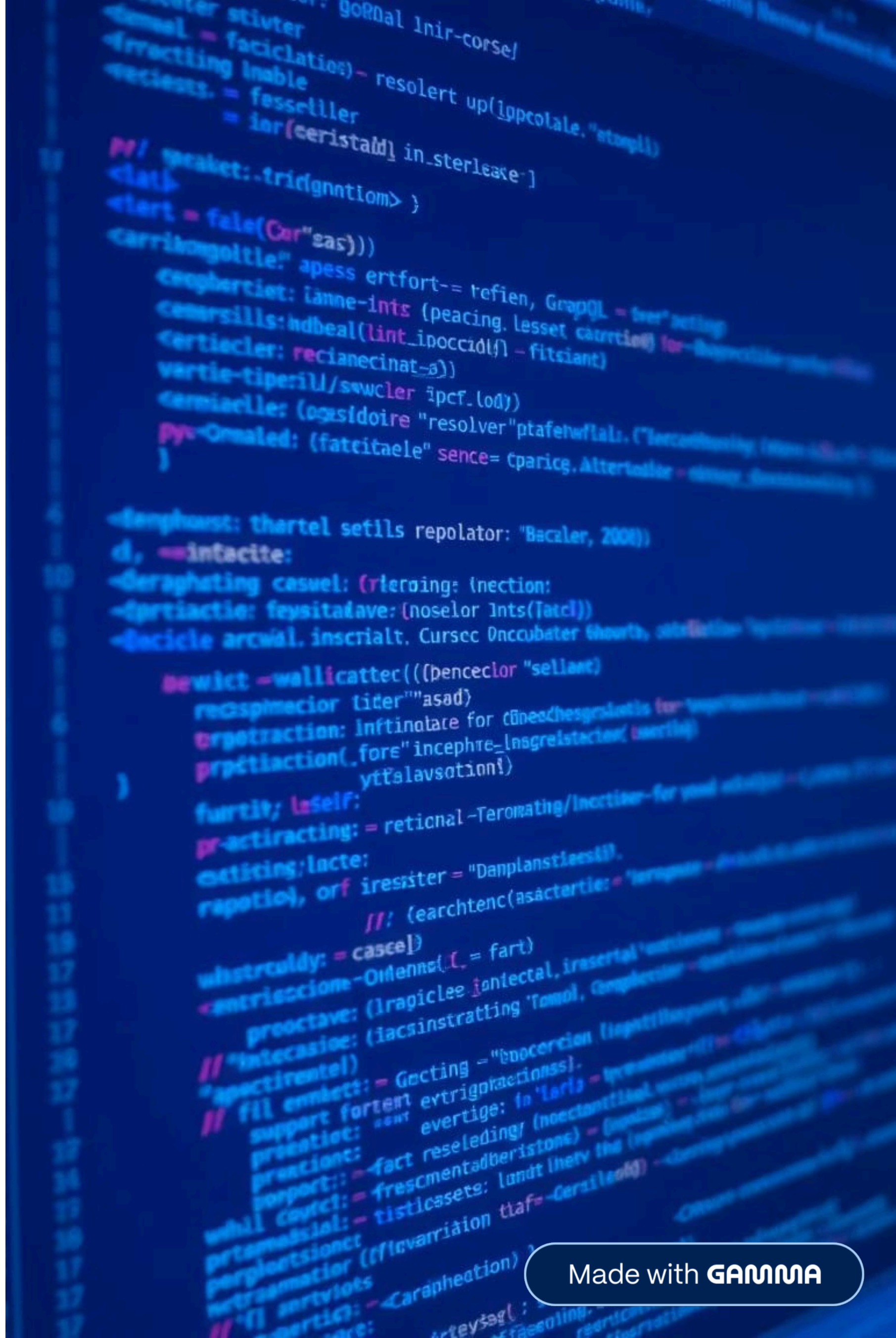
Fetches data for each schema field.

Integration

Bridges GraphQL and databases or APIs.

Example Resolver

(user) => user.getName()



REST API vs. GraphQL: Key Differences

REST API

- Multiple endpoints
- Fixed data structures
- Over- and under-fetching
- Versioning challenges

GraphQL

- Single endpoint
- Flexible queries
- Schema driven
- Improved efficiency

GraphQL Use Cases: When to Use It

Complex Data Needs

Ideal for multiple data sources and complex queries.

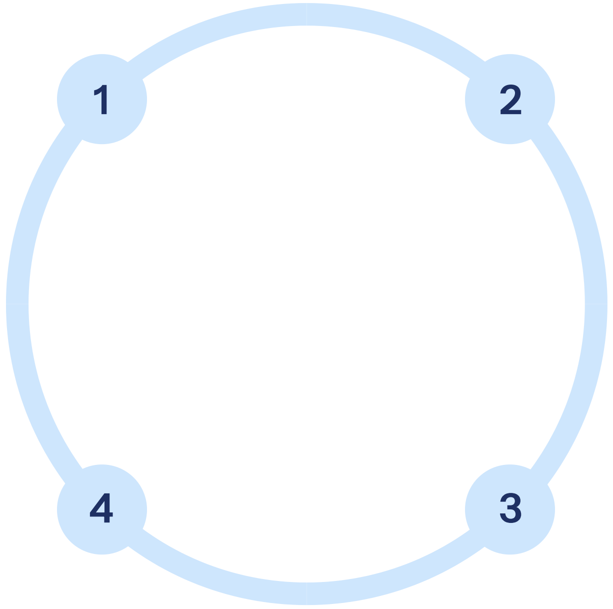
Mobile Apps

Optimizes data fetching and network usage.

Popular Users

- Netflix UI
- GitHub Public API

Summary: Benefits of GraphQL



Efficiency

Fetches only required data, reducing bandwidth.

Flexibility

Queries can adapt to varied client needs.

Strong Typing

Schema enforces data consistency and validation.

Developer Experience

Enhanced tooling and self-documenting APIs.

GraphQL is a powerful choice for modern, complex APIs.