GOING GRAPHQL

@CHRISBISCARDI

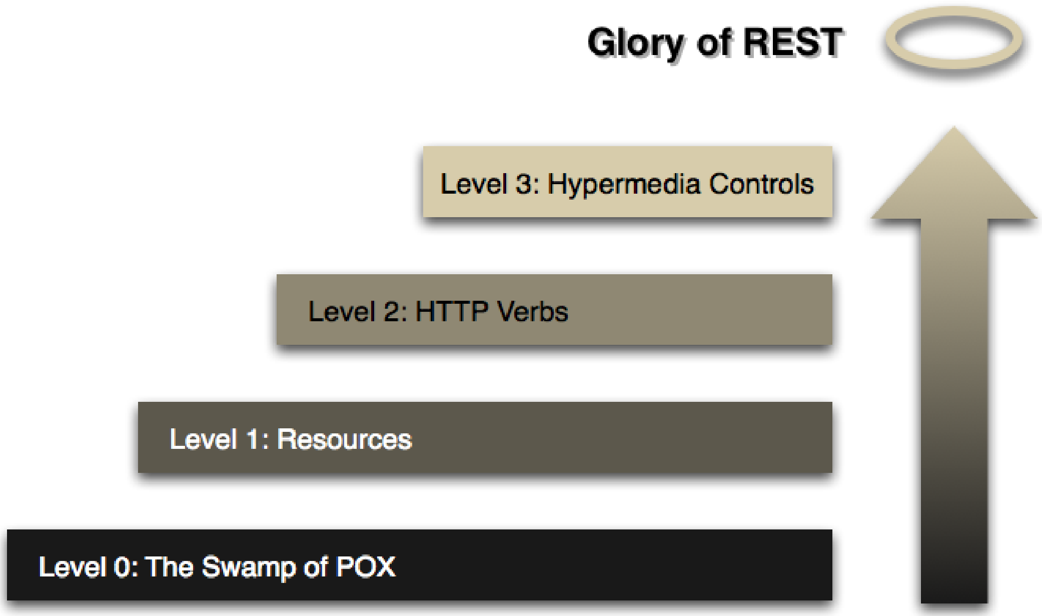
- Docker
- GraphQL
- Consulting

APIS FOR WHO?

BUILT WITH WHAT?

- REST
- GRPC
- GraphQL

REST



Glory of REST

Level 3: Hypermedia Controls

Level 2: HTTP Verbs

Level 1: Resources

Level 0: The Swamp of POX

URL STRUCTURE

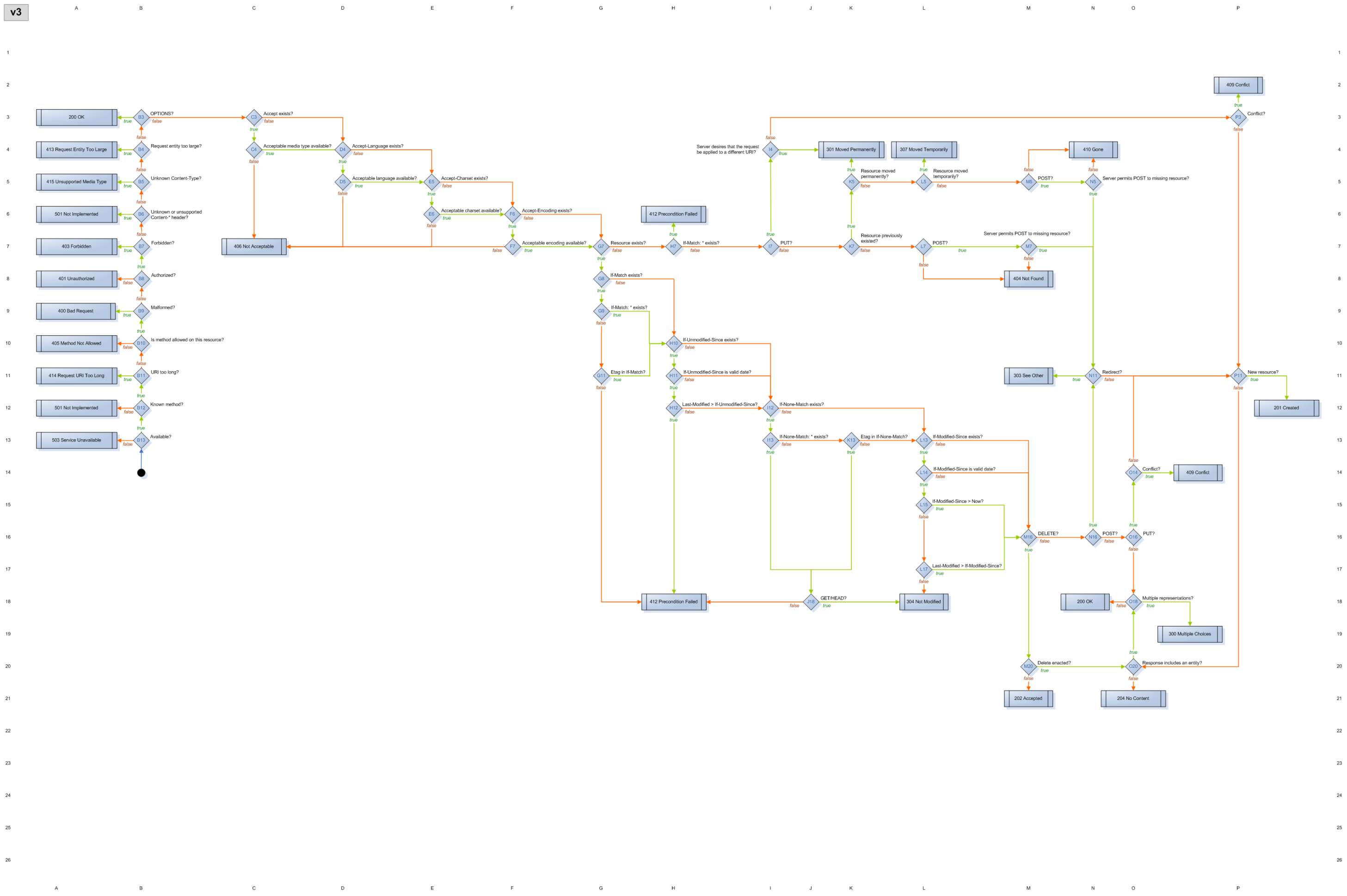
```
authors/

authors/1

authors/1/books

authors/1/books/10

authors/1/books/10/summary
```

HTTP

HTTP/1.1 (DELETE, GET, HEAD, PUT, POST)

An activity diagram to describe the resolution of the response status code, given various headers

Creator: http://thoughtpad.net/alan-dean
Source: http://thoughtpad.net/alan-dean/http-headers-status


v3
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
200 OK
B3 OPTIONS?
C3 Accept exists?
413 Request Entity Too Large
B4 Request entity too large?
C4 Acceptable media type available?
D4 Accept-Language exists?
415 Unsupported Media Type
B5 Unknown Content-Type?
D5 Acceptable language available?
E5 Accept-Charset exists?
501 Not Implemented
B6 Unknown or unsupported Content-* header?
E6 Acceptable charset available?
F6 Accept-Encoding exists?
403 Forbidden
B7 Forbidden?
406 Not Acceptable
F7 Acceptable encoding available?
G7 Resource exists?
401 Unauthorized
B8 Authorized?
G8 If-Match exists?
400 Bad Request
B9 Malformed?
G9 If-Match: * exists?
405 Method Not Allowed
B10 Is method allowed on this resource?
414 Request URI Too Long
B11 URI too long?
G11 Etag in If-Match?
501 Not Implemented
B12 Known method?
503 Service Unavailable
B13 Available?
409 Conflict
P3 Conflict?
I4 Server desires that the request be applied to a different URI?
301 Moved Permanently
307 Moved Temporarily
410 Gone
K5 Resource moved permanently?
L5 Resource moved temporarily?
M5 POST?
N5 Server permits POST to missing resource?
412 Precondition Failed
H7 If-Match: * exists?
I7 PUT?
K7 Resource previously existed?
L7 POST?
M7 Server permits POST to missing resource?
404 Not Found
H10 If-Unmodified-Since exists?
H11 If-Unmodified-Since is valid date?
303 See Other
N11 Redirect?
P11 New resource?
201 Created
H12 Last-Modified > If-Unmodified-Since?
I12 If-None-Match exists?
I13 If-None-Match: * exists?
K13 Etag in If-None-Match?
L13 If-Modified-Since exists?
L14 If-Modified-Since is valid date?
O14 Conflict?
409 Conflict
L15 If-Modified-Since > Now?
M16 DELETE?
N16 POST?
O16 PUT?
L17 Last-Modified > If-Modified-Since?
412 Precondition Failed
J18 GET/HEAD?
304 Not Modified
200 OK
O18 Multiple representations?
300 Multiple Choices
M20 Delete enacted?
O20 Response includes an entity?
202 Accepted
204 No Content
true
false
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

Documentation

Overfetching

Underfetching (N+1)

Backends for Frontends

GRPC

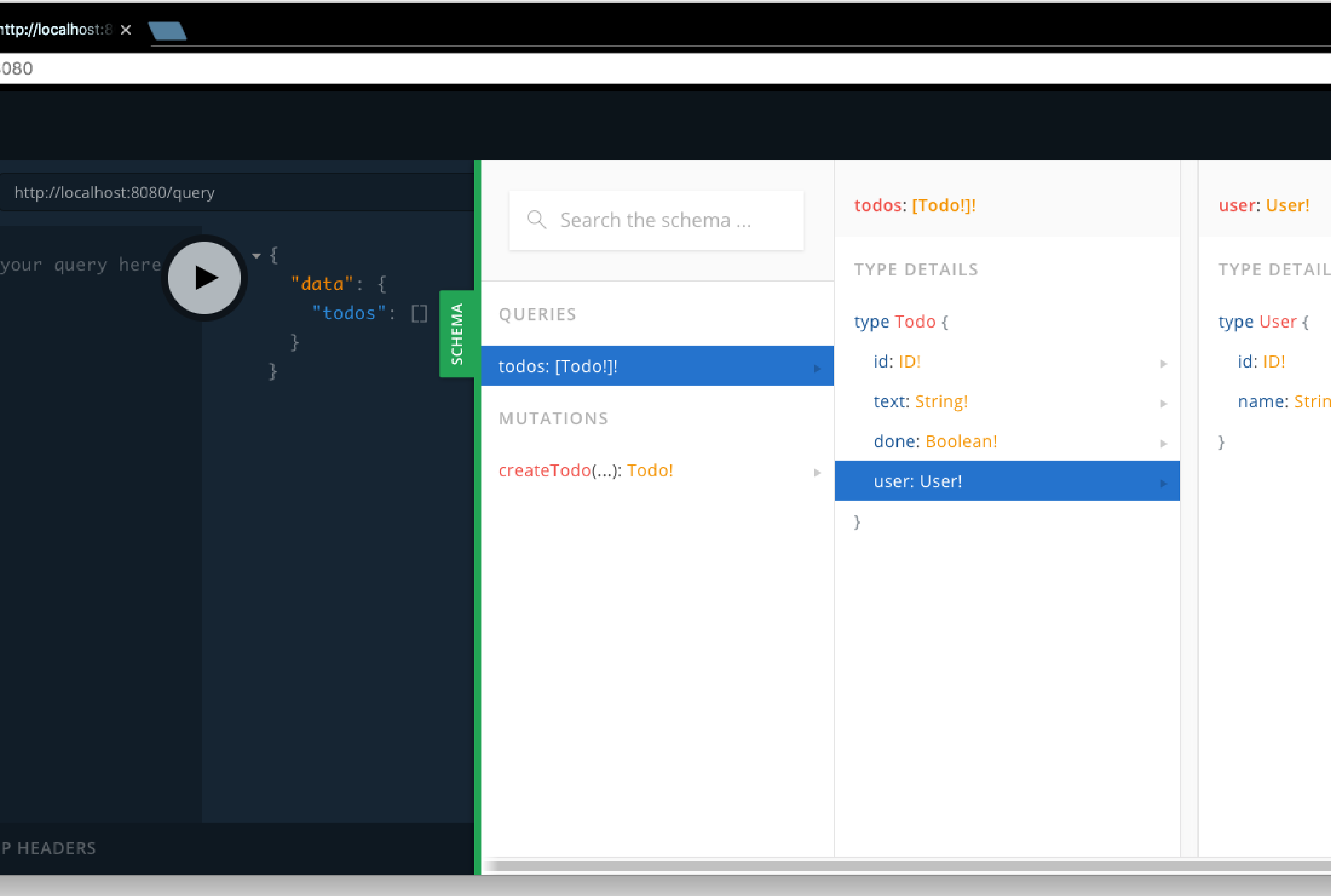
- Authentication
- Bidirectional streaming and flow control
- Blocking or nonblocking bindings
- Cancellation
- Timeouts

GRPC Web

WHY GRAPHQL?

PRODUCT APIS

DOCUMENTATION



```
your query here
```

```
{
  "data": {
    "todos": []
  }
}
```

todos: [Todo!]!

createTodo(...): Todo!

todos: [Todo!]!

```
type Todo {
  id: ID!
  text: String!
  done: Boolean!
  user: User!
}
```

user: User!

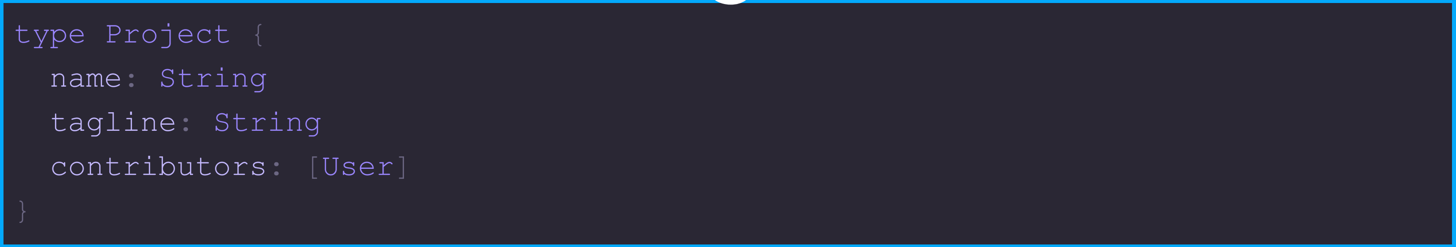
```
type User {
  id: ID!
  name: Strin
}
```

INTRO TO GRAPHQL

WHAT IS GRAPHQL

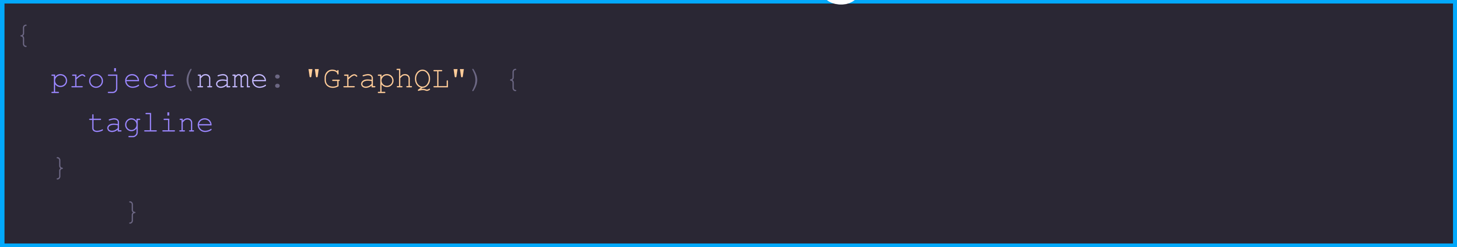
- Query Language
- Types vs Endpoints
- Powerful Devtools
- API Evolution

Describe your Data



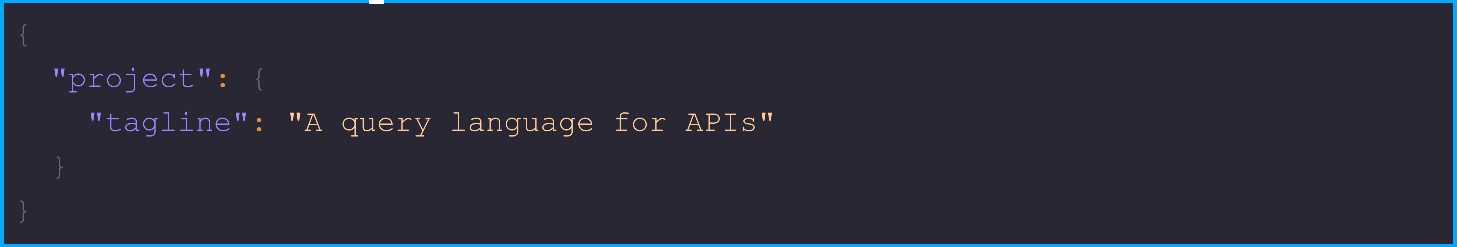
```
type Project {
  name: String
  tagline: String
  contributors: [User]
}
```

Ask for what you want



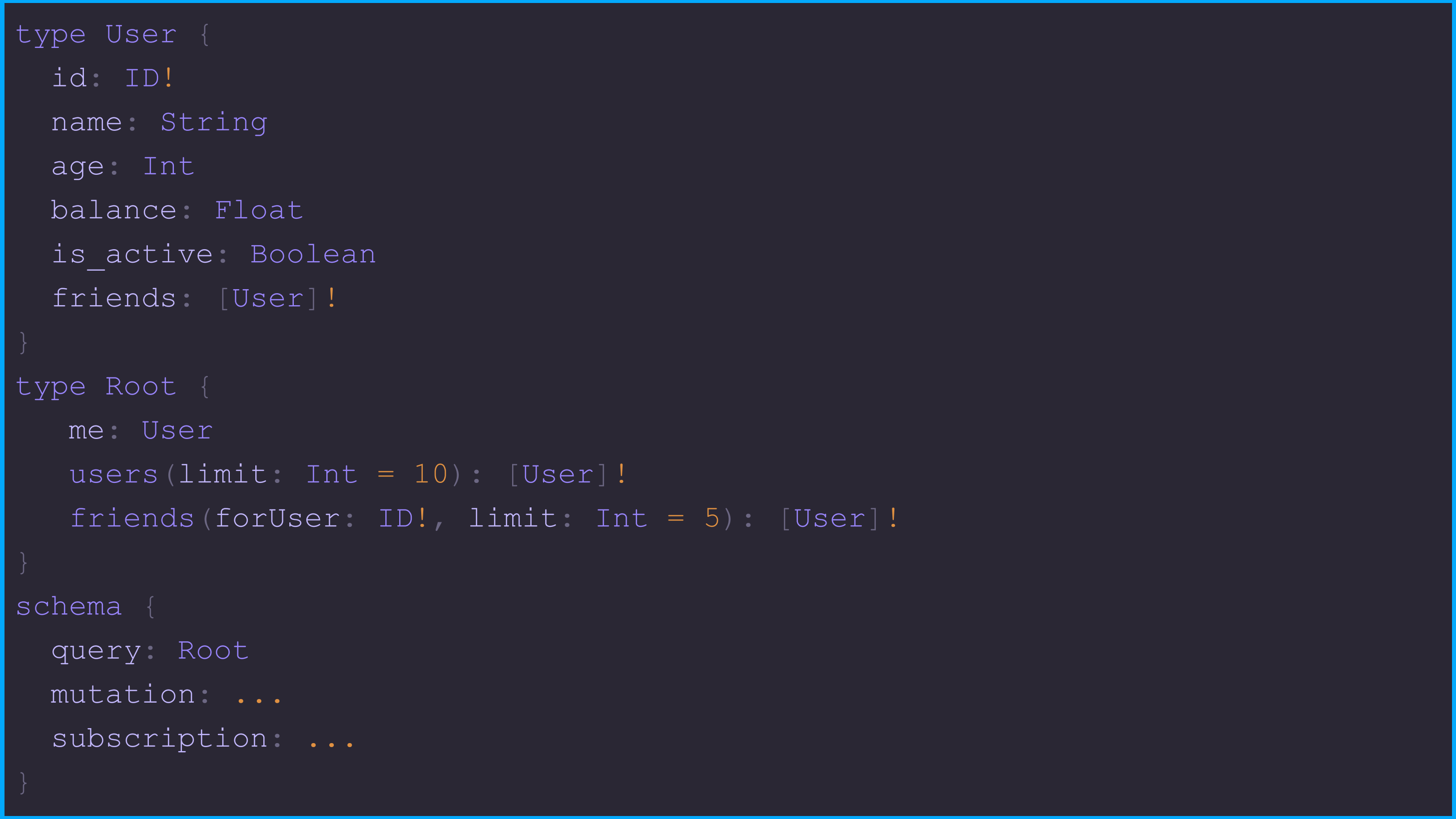
```
{
  project(name: "GraphQL") {
    tagline
  }
      }
```

Get predictable results



```
{
  "project": {
    "tagline": "A query language for APIs"
  }
}
```

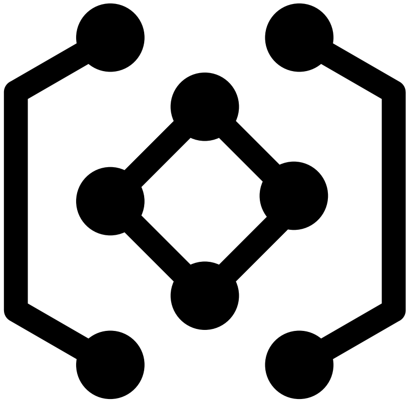
SDL



```
type User {
  id: ID!
  name: String
  age: Int
  balance: Float
  is_active: Boolean
  friends: [User]!
}
type Root {
   me: User
   users(limit: Int = 10): [User]!
   friends(forUser: ID!, limit: Int = 5): [User]!
}
schema {
  query: Root
  mutation: ...
  subscription: ...
}
```

GQLGEN

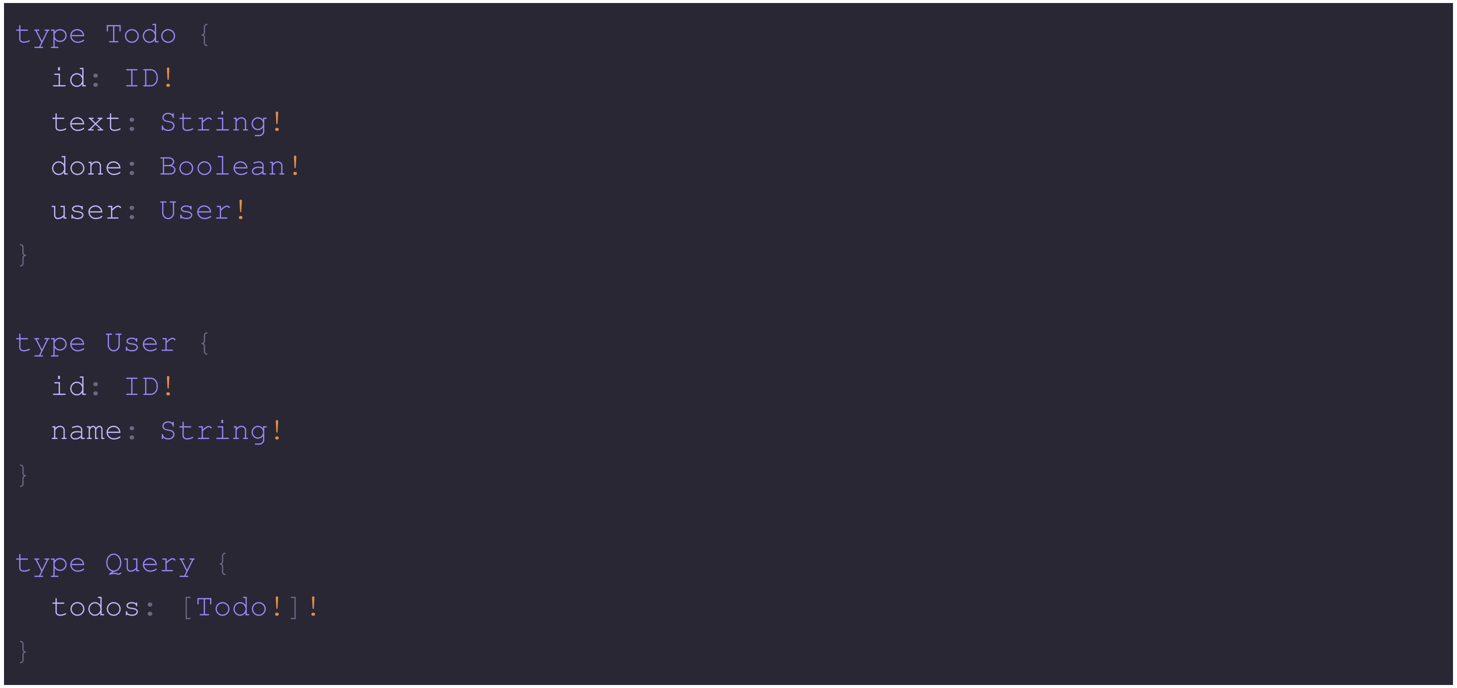
99designs



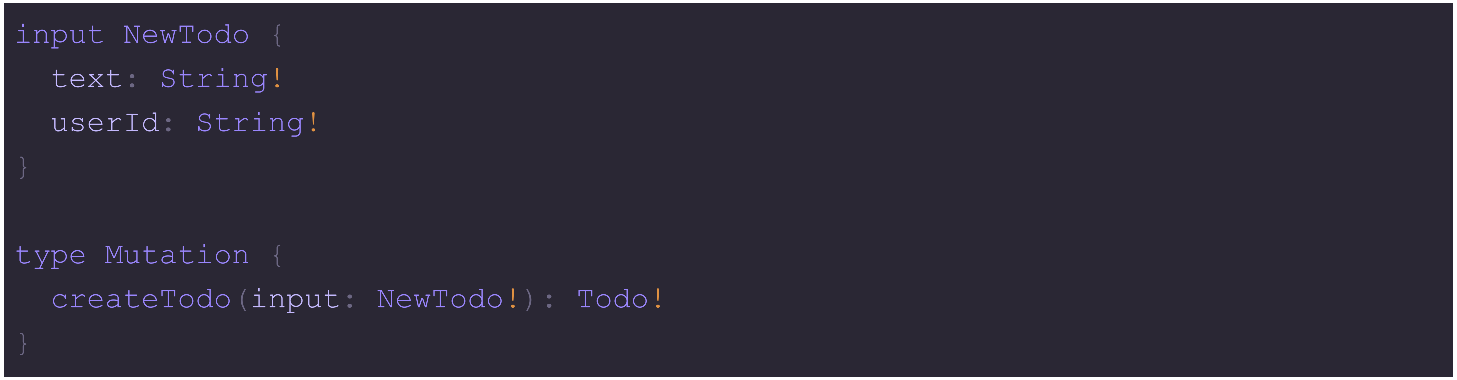
WHAT IS GQLGEN?

- Schema first
- Type safe
- Codegen

```
go get -u github.com/99designs/gqlgen \
          github.com/vektah/gorunpkg
```



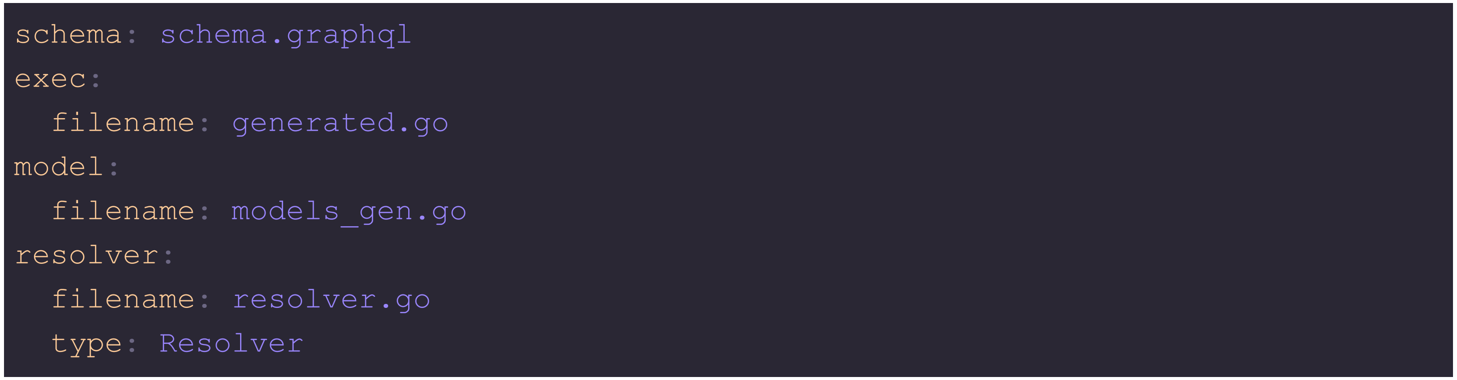
```graphql
type Todo {
  id: ID!
  text: String!
  done: Boolean!
  user: User!
}

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

type Query {
  todos: [Todo!]!
}
```



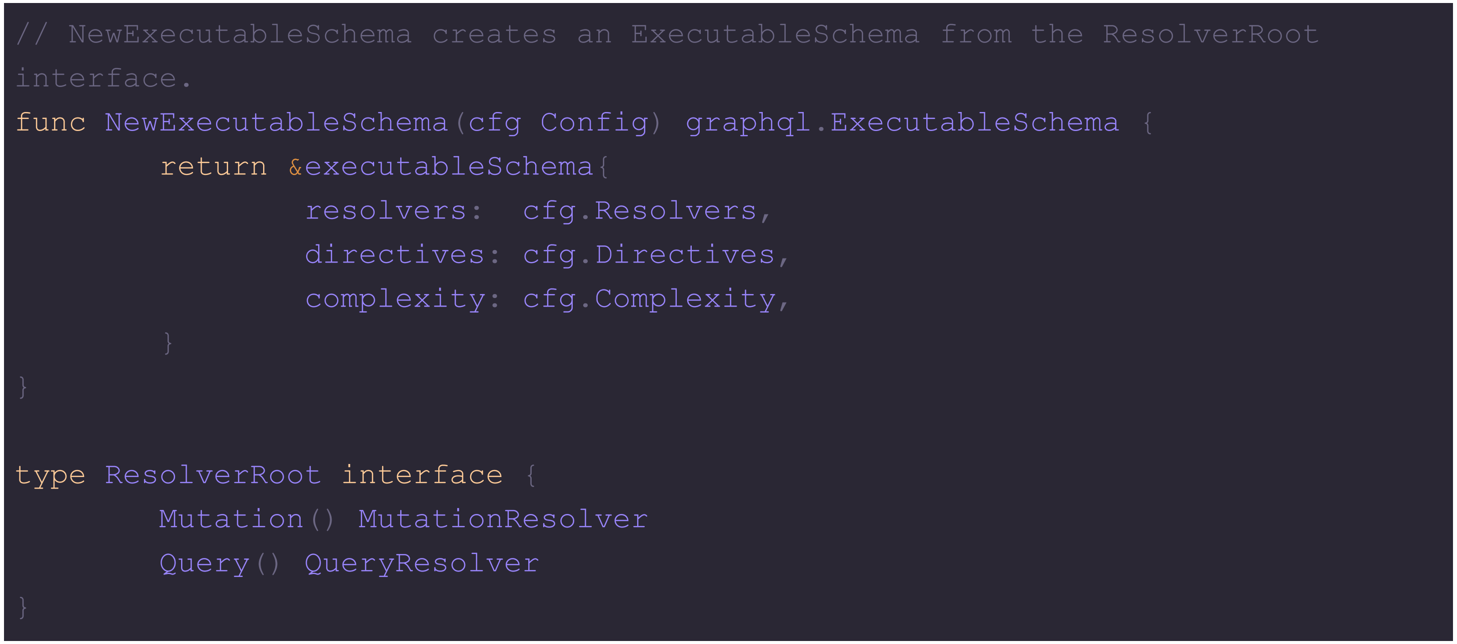
```graphql
input NewTodo {
  text: String!
  userId: String!
}

type Mutation {
  createTodo(input: NewTodo!): Todo!
}
```

```go
package model

type User struct {
  ID   string
  Name string
}
```

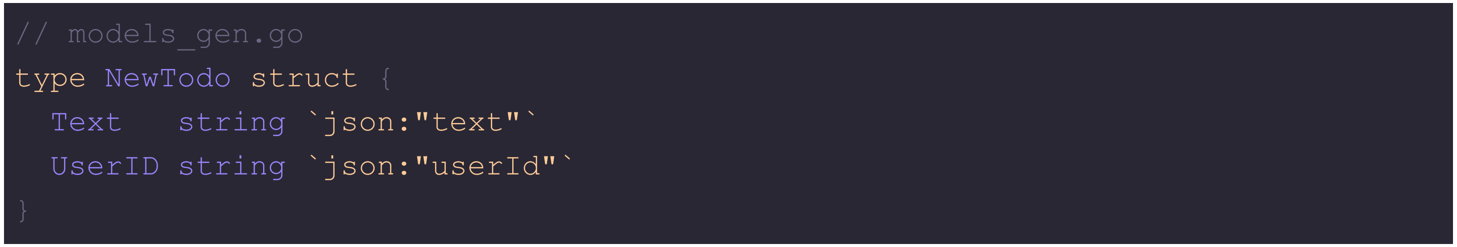
```
gqlgen init
```



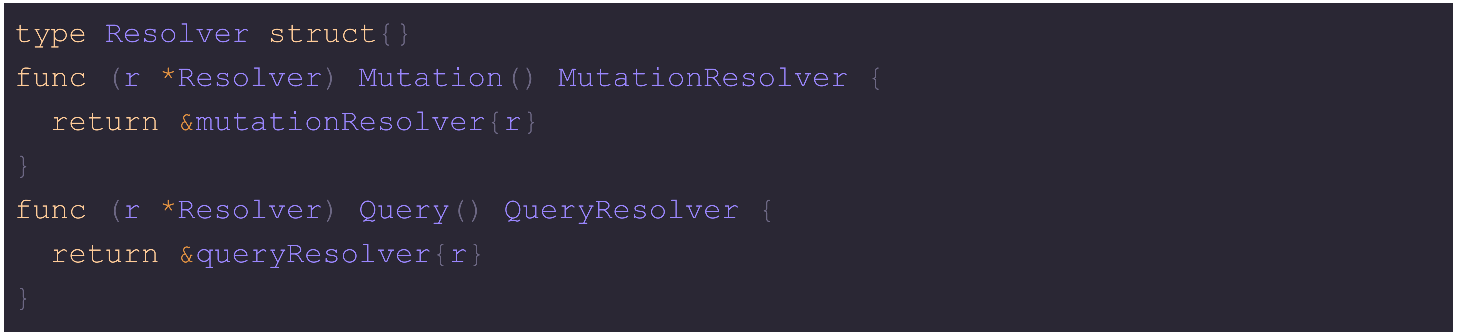
```yaml
schema: schema.graphql
exec:
  filename: generated.go
model:
  filename: models_gen.go
resolver:
  filename: resolver.go
  type: Resolver
```



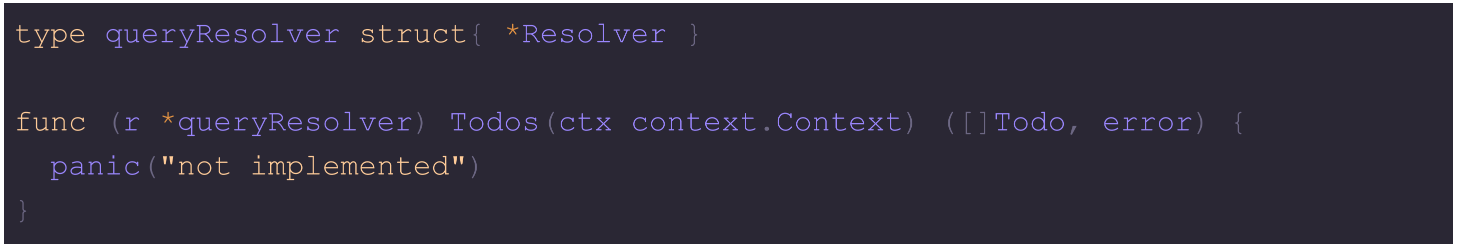
```
// NewExecutableSchema creates an ExecutableSchema from the ResolverRoot
interface.
func NewExecutableSchema(cfg Config) graphql.ExecutableSchema {
        return &executableSchema{
                resolvers:  cfg.Resolvers,
                directives: cfg.Directives,
                complexity: cfg.Complexity,
        }
}

type ResolverRoot interface {
        Mutation() MutationResolver
        Query() QueryResolver
}
```



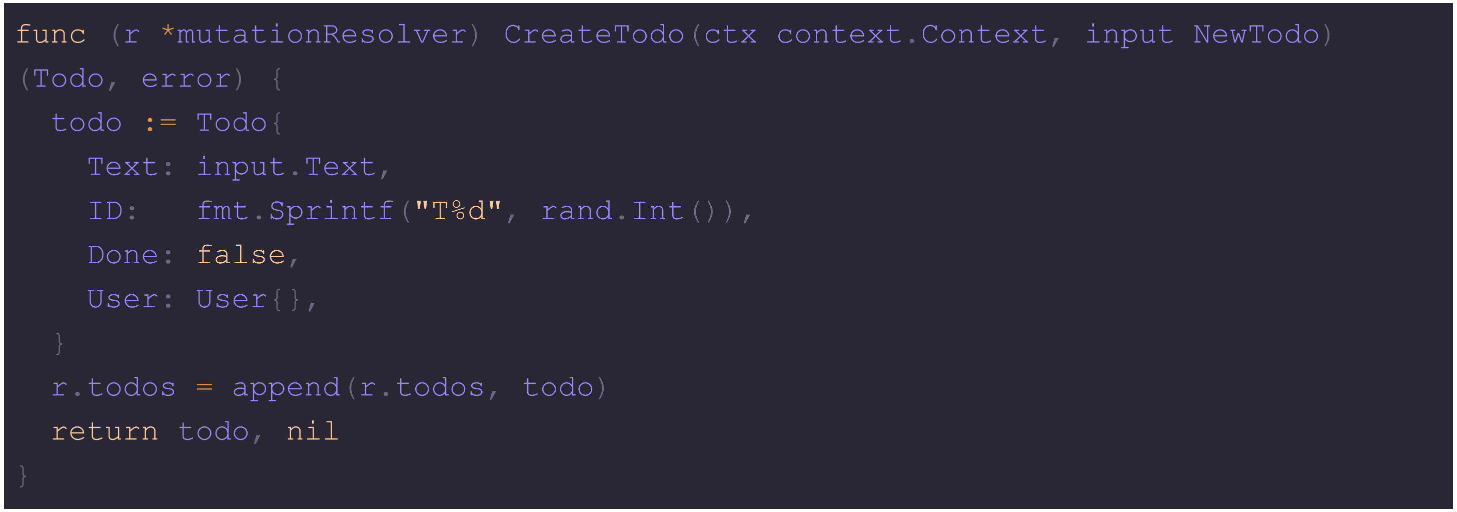
```go
// models_gen.go
type NewTodo struct {
  Text   string `json:"text"`
  UserID string `json:"userId"`
}
```



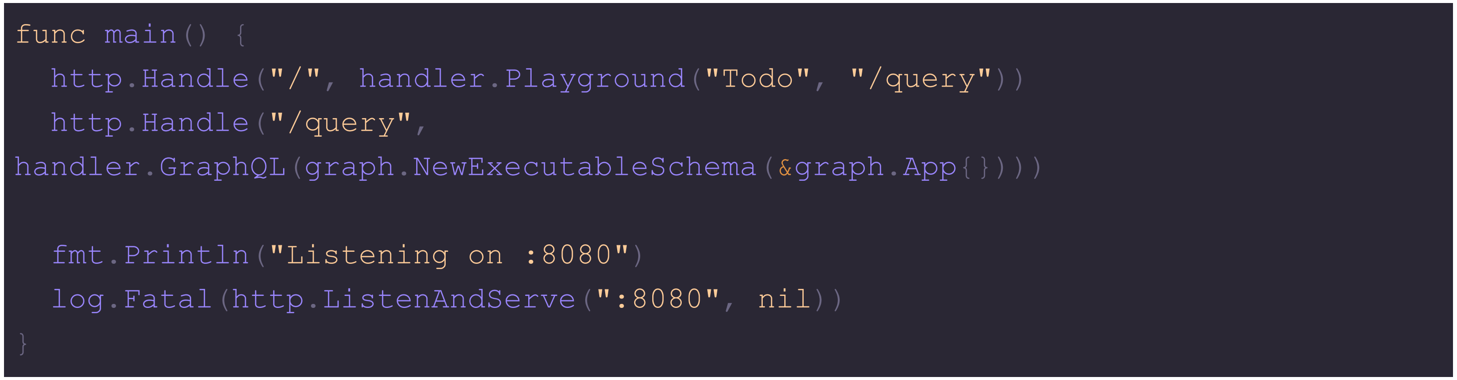
```
type Resolver struct{}
func (r *Resolver) Mutation() MutationResolver {
  return &mutationResolver{r}
}
func (r *Resolver) Query() QueryResolver {
  return &queryResolver{r}
}
```



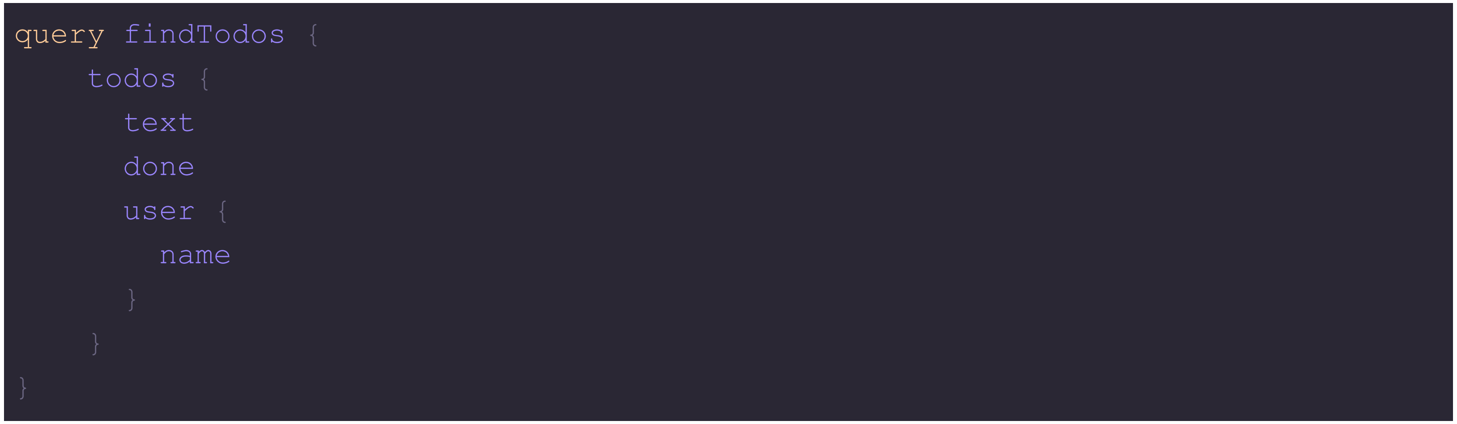
```go
type queryResolver struct{ *Resolver }

func (r *queryResolver) Todos(ctx context.Context) ([]Todo, error) {
  panic("not implemented")
}
```



```go
func (r *mutationResolver) CreateTodo(ctx context.Context, input NewTodo)
(Todo, error) {
  todo := Todo{
    Text: input.Text,
    ID:   fmt.Sprintf("T%d", rand.Int()),
    Done: false,
    User: User{},
  }
  r.todos = append(r.todos, todo)
  return todo, nil
}
```



```go
func main() {
  http.Handle("/", handler.Playground("Todo", "/query"))
  http.Handle("/query",
handler.GraphQL(graph.NewExecutableSchema(&graph.App{})))

  fmt.Println("Listening on :8080")
  log.Fatal(http.ListenAndServe(":8080", nil))
}
```



```
query findTodos {
    todos {
      text
      done
      user {
        name
      }
    }
}
```

0.6

- sponsorship (99designs)
- June 2018 GraphQL Spec
- Directives (@skip, @limit, custom)
- Query complexity
- Can generate resolver stubs

EXTRAS

DATALOADERS

```
query { todos { users { name } } }
```

```go
func (r *Resolver) Todo_user(ctx context.Context, obj *Todo) (*User, error) {
  res := logAndQuery(r.db, "SELECT id, name FROM user WHERE id = ?", obj.UserID)
  defer res.Close()

  if !res.Next() {
    return nil, nil
  }
  var user User
  if err := res.Scan(&user.ID, &user.Name); err != nil {
    panic(err)
  }
  return &user, nil
}
```