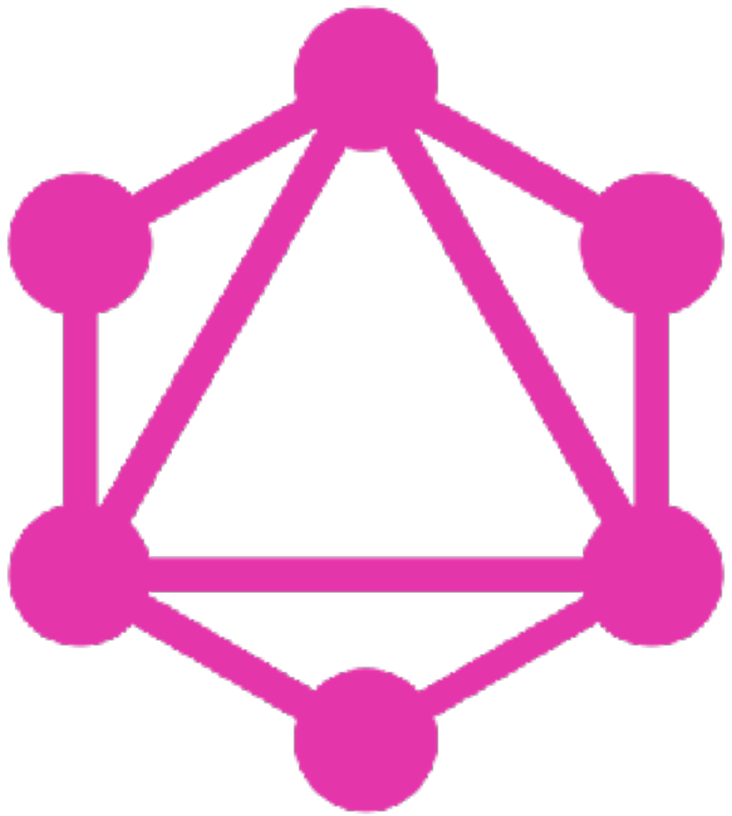


GraphQL til folket

Seán Erik Scully © 2017

Hva er GraphQL?

Et spørrespråk (query language)

- Ikke et databasespørrespråk
- Alternativ til REST
- Tatt i bruk av Facebook siden 2012
- Open source i 2015

REST

- Logisk eksponering av ressurser
- Bruker HTTP-metoder
- Tilstandsløst => Enkelt å skalere
- Støtter innhold på flere formater
- Lett å forstå og enkelt å teste

- **GET /events - Retrieves a list of events**
- **GET /events/1 - Retrieves a specific events**
- **POST /events - Creates a new event**
- **PUT /events/1 - Updates event #1**
- **PATCH /events/1 - Partially updates event #1**
- **DELETE /events/1 - Deletes event #1**

Utfordringer

- Henter unødvendig informasjon
- Mange kall => ett (aggregert) endepunkt for visning
- Du har mistet kontroll over API-et ditt!
- Versjonering?
- Må oppdatere dokumentasjon!
- REST? => ad hoc-endepunkter

GraphQL

Operasjoner

Query => Spør om data

Mutation => Endre data

GraphQL og query

```
{
  events {
    id
    name
  }
}
```

```
{
  "events": [
    {
      "id": "1",
      "name": "Event no 1"
    },
    {
      "id": "2",
      "name": "Event no 2"
    },
    {
      "id": "3",
      "name": "Event no 3"
    }
  ]
}
```

Query

```
{
  events(id: "1") {
    name
  }
}
```

```
{
  "events":
    {
      "name": "Event no 1"
    }
}
```

Query

```
{
  human(id: "1000") {
    name
    height(unit: FOOT)
  }
}
```

```
{
    "human": {
      "name": "Luke Skywalker",
      "height": 5.6430448
    }
}
```

Mutation

```
{
  createEvent(name: "Party")
{
    id
    name
  }
}
```

```
{
    createEvent {
      id:"1"
      name: "Party"
    }
}
```

Variabler

```
mutation createEvent($name: String!) {
  createEvent(name: $name) {
    id
  }
}
```

Fragments

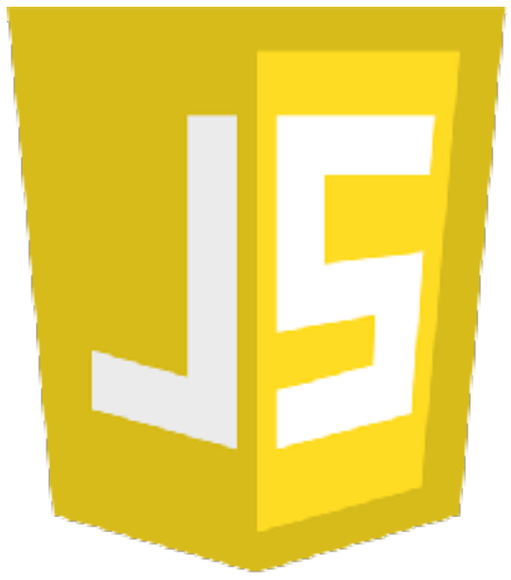
```
{
  stopPlace(id: "NSR:StopPlace:8") {
    ...myStop
  }

}

fragment myStop on StopPlace {
  name {
    value
    lang
  }
  geometry {
    type
    coordinates
  }
  stopPlaceType
}
```

Demo-tid

GraphQL på server

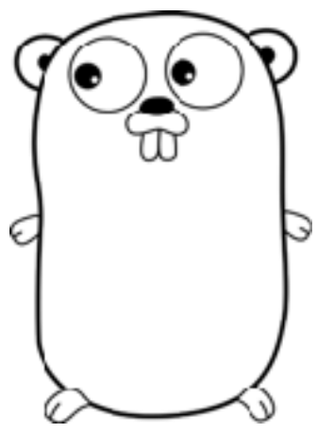
- Protokoll
- API-lag
- GraphQL-Implementasjon
- Schema
- Datalag

Implementasjoner



graphql-js

graphql-core + graphene



graphql-go

graphql-ruby

sangria

graphql-java

Server-implementasjon

- Lexer / parser: Query => AST (Abstract Syntax Tree)
- Type System => Støtte for å definere typer og schema
- Introspection
- Validering mot “Type system”
- Execution

GraphQL på klient

- Programmeringsspråk velger du selv
- Protokoll velger du selv
- Begrenset av serverens schema
- En GraphQL-klient!

Lokka

Relay

Hva gir klienten deg?

- Enkelt å bytte ut nettverkslaget
- Caching
- Refetching / synkronisering
- Løser problemene for deg!

HTTP

POST https://localhost/graphql

payload query, variabler, …

Klient → Server

HTTP

OK 200

Klient

```
{
  "data": {
    "stopPlace": [
      {
        "id": "NSR:StopPlace:1"
      }
    ]
  }
}
```

Server

HTTP

Bad request 400

```
{
  "errors": [
    "Validation error of type FieldUndefined: Field tullefelt is undefined"
  ]
}
```

Klient

Server

Schema og typer

- Hva er tilgjengelig
- På hvilket format - typestrengt

```
schema {
  query: Query
}

type Query {
  hello: String
}
```

```
var schema = buildSchema(`
  type Query {
    hello: String
  }
`);

var root = { hello: () => 'Hello world!' };

graphql(schema, '{ hello }', root).then((response)
=> {
  console.log(response); // { data: { hello: 'Hello world!' } }
});
```

```
GraphQLObjectType queryType = newObject()
                                .name("helloWorldQuery")
                                .field(newFieldDefinition()
                                        .type(GraphQLString)
                                        .name("hello")
                                        .staticValue("world"))
                                .build();

        GraphQLSchema schema = GraphQLSchema.newSchema()
                                .query(queryType)
                                .build();
        Map<String, Object> result = new
GraphQL(schema).execute("{hello}").getData();

        System.out.println(result); //{hello=world}
```

Scalar

- Int
- Float
- String
- Boolean
- ID

Enum

```
enum AnimalType {
  Cat
  Dog
  Horse
}

type Child {
  name: String
  pet: AnimalType
}
```

Interface

```
interface Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
}
```

```
type Human implements Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
  starships: [Starship]
  totalCredits: Int
}
```

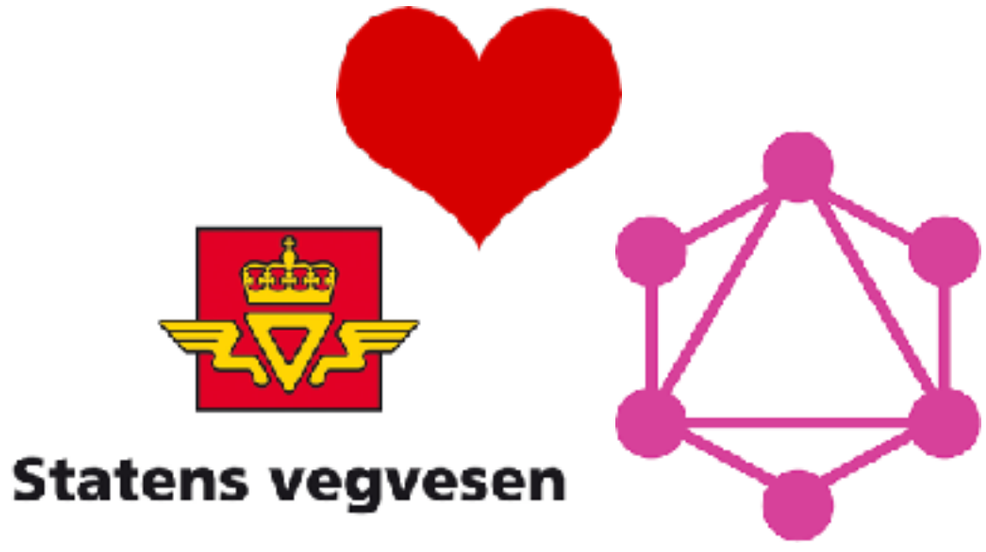

LET'S BE HONEST HERE
TYPES ARE USEFUL
memegenerator.net

Når trenger du GraphQL?

- Når hastighet og kort ventetid teller
- Du har mange klienter (apper, websider …)
- Når data er sentralisert og eksponert ett sted
- Dokumentasjon og versjonering er viktig

GraphQL i Statens Vegvesen

- Digitransit-UI / www.dit.no
- Nasjonalt Stoppestedsregister



Come an!

- Nanokurs i GraphQL - https://learngraphql.com/
- God oversikt - https://github.com/chentsulin/awesome-graphql
- Den offisielle siden - http://graphql.org/
- GraphQL Europe, Twitter og Github!

Tusen takk!

Spørsmål?