Lower Latency with Streaming GraphQL - @defer & @stream

GraphQL Conf
2026

Rob Richard @rob_richard

1stdibs.com

Enjoy Free Shipping on Art, Fashion & Decor over $1,000 with code FREESHIP. Terms apply. Details

1stDIBS

Search 1stDibs

Log In | Sign Up

NEW ARRIVALS FURNITURE LIGHTING ART JEWELRY WATCHES FASHION WORLD OF 1STDIBS SALE

Home / Furniture

Furniture

SHOP NOW

Furniture under $2,500

SHOP NOW

Art Deco Pieces

NEW & NOTABLE

The Latest Lighting

Popular Now

Mid-Century Modern Fu...

Shop All

Cassina Furniture

Shop All

Knoll Furniture

Shop All

Shop Furniture

What problem are we trying to solve?

- Major benefit of GraphQL
  - Get many resources in a single request
- Drawback of this
  - Your page loads as fast as its slowest piece of data
- Defer and stream allows you to deprioritize parts of your query to lower latency of the most important parts

What is @defer and @stream?

- New directives for GraphQL
- The goal is to prioritize the delivery of essential data
- Client tells GraphQL server which parts of the query can be deferred
- Allows the non-deferred data to be returned to clients sooner

```
query {
  person(id: "1") {
    ...HomeWorldFragment @defer(label: "homeWorldDefer")
    name
    films @stream(initialCount: 2, label: "filmsStream") {
      title
    }
  }
}
fragment HomeWorldFragment on Person {
  homeworld {
    name
  }
}
```

@defer

```
directive @defer(if: Boolean! = true, label: String) on FRAGMENT_SPREAD | INLINE_FRAGMENT
```

- The **@defer** directive may be specified on a fragment spread or inline fragment.
- **if: Boolean! = true**
  - When **true** fragment may be deferred, defaults to **true**.
- **label: String**
  - A unique label across all **@defer** and **@stream** directives in an operation.

@stream

```
directive @stream(if: Boolean! = true, label: String, initialCount: Int) on FIELD
```

- The **@stream** directive may be provided for a field of **List**
- **if: Boolean = true**
    - When **true** list items are streamed, defaults to **true**.
- **label: String**
    - A unique label across all **@defer** and **@stream** directives in an operation.
- **initialCount: Int**
    - The number of list items the server should return as part of the initial payload.

Demo

Why @defer and @stream?

- Why not multiple queries?
  - Figuring out how to chop up your query into different pieces and coordinating those requests on the client.
  - Additional roundtrip cost of making separate GraphQL queries.
  - N+1, Later queries may require input from previous queries

Why @defer and @stream?

- Why not subscriptions?
  - Subscriptions communicate real time events
  - Defer and stream prioritize the delivery of essential data
  - Results can be delivered over a short lived chunked http connection
  - Uses simple HTTP infrastructure, no websockets or stateful connections required.

Current status of @defer and @stream

- First proposed to the GraphQL WG based on Facebook implementation
- New response format developed
  - Current protocol has been stable for 2+ years
  - Implemented in GraphQL-JS v17.0.0 beta and others
  - Spec edits are written and under final review by the GraphQL WG:
    - https://github.com/graphql/graphql-spec/pull/1110

Can I use?

YES! https://caniuse.graphql.now/feature/defer.html

Servers:

- GraphQL-JS v17.0.0-beta.1
- Apollo Server v5.1.0
- Hot Chocolate v16

Gateways:

- GraphQL Fusion v16

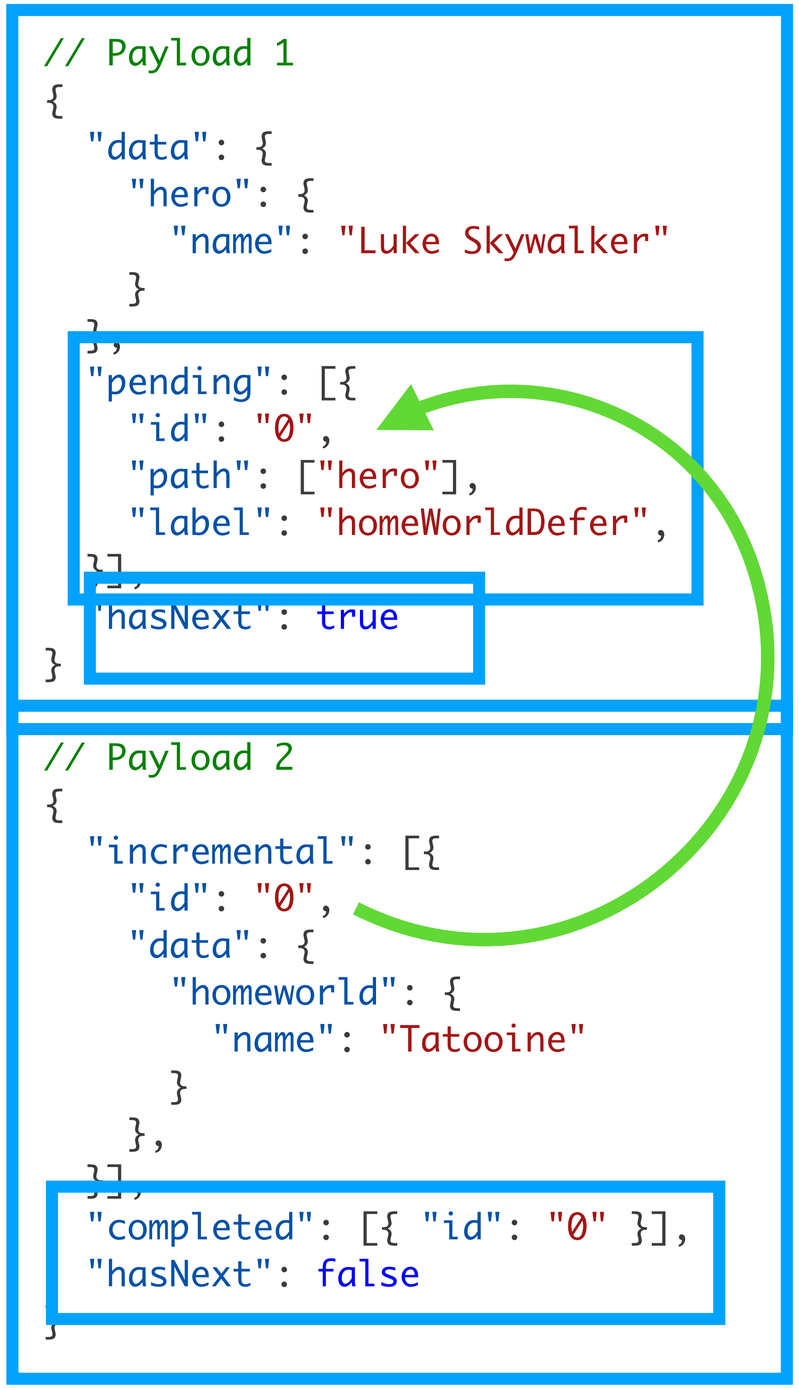
Clients:

- Apollo Client v4.1
- Apollo Kotlin v5.0.0
- Urql v2.3.0
- Relay (main branch, requires adapter)

@defer Response Format

- **pending** — @defer at given path with label has been assigned an id and will be delivered incrementally
- **hasNext** — A boolean that is present and true when there are more responses that will be sent for this operation.
- **incremental** — The data that is being delivered incrementally.
- **completed** —all the data for the given id has been delivered

```
query {
  hero(id: "1") {
    ...HomeWorldFragment @defer(label: "homeWorldDefer")
    name
  }
}
fragment HomeWorldFragment on Person {
  homeworld {
    name
  }
}
```



@stream Response Format

- **pending** — @stream at given path with label has been assigned an id and additional list items will be delivered incrementally
- **hasNext** — A boolean that is present and true when there are more responses that will be sent for this operation.
- **incremental** —new list items to append to the streamed list
- **path** — A list of keys from the root of the response to the insertion point
- **completed** —all the data for the given id has been delivered

```
query {
  hero(id: "1") {
    name
    films @stream(initialCount: 1, label: "filmsStream") {
      title
    }
  }
}
```

```
// Payload 1
{
  "data": {
    "hero": {
      "films": [{ "title": "A New Hope" }]
    }
  },
  "pending": [{
    "id": "0",
    "label": "filmsStream",
    "path": ["hero", "films"]
  }],
  "hasNext": true
}
```

```
// Payload 2
{
  "incremental":[{
    "id": "0",
    "items": [{"title": "The Empire Strikes Back"}]
  }],
  "hasNext": true
}
```

```
// Payload 3
{
  "incremental":[{
    "id": "0",
    "items": [{"title": "Return of the Jedi"}]
  }],
  "completed": [{ "id": "0" }],
  "hasNext": false
}
```

Older protocols

- The response format has evolved over time as we've been developing this feature
- If you're using @defer or @stream now, you may be using an older protocol
  - graphql-js@16.1.0-experimental-stream-defer.6

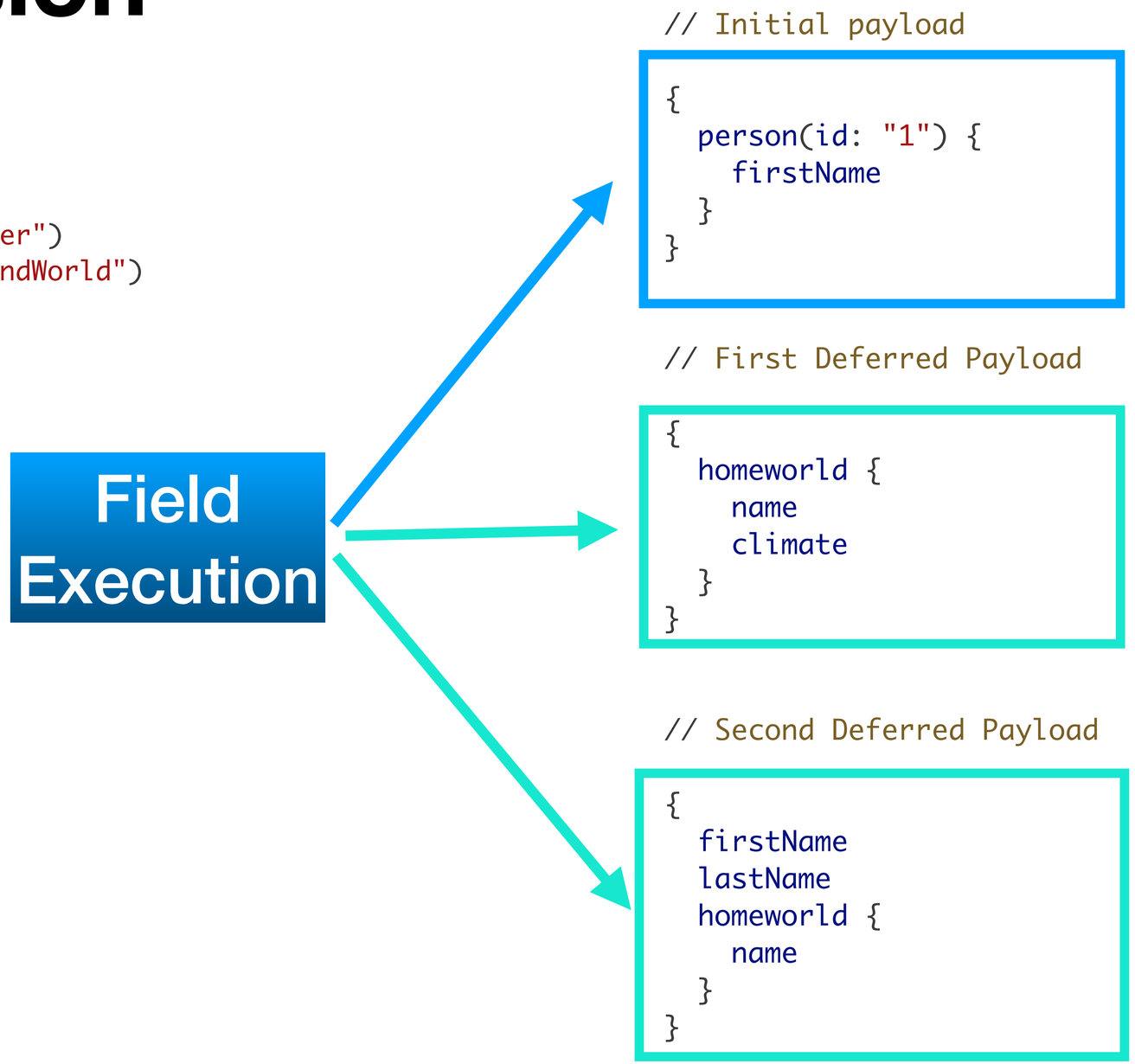
You should upgrade!

- By upgrading to the latest spec proposal your application might see major performance improvements
- The latest proposal ensures fields are only executed once, even if they are included in both deferred and non-deferred fragments
- Previous iterations would cause resolvers to be executed multiple times.
- Latest spec: GraphQL-JS v17.0.0-alpha.7 or later
- Transform modern format to legacy format: @yaacovcr/transform

Example - old version

```
query {
  person(id: "1") {
    ...HomeWorldFragment @defer(label: "homeWorldDefer")
    ...NameAndHomeWorldFragment @defer(label: "nameAndWorld")
    firstName
  }
}
fragment HomeWorldFragment on Person {
  homeworld {
    name
    climate
  }
}

fragment NameAndHomeWorldFragment on Person {
  firstName
  lastName
  homeworld {
    name
  }
}
```



Example - old version

```
query {
  person(id: "1") {
    ...HomeWorldFragment @defer(label: "homeWorldDefer")
    ...NameAndHomeWorldFragment @defer(label: "nameAndWorld")
    firstName
  }
}
fragment HomeWorldFragment on Person {
  homeworld {
    name
    climate
  }
}

fragment NameAndHomeWorldFragment on Person {
  firstName
  lastName
  homeworld {
    name
  }
}
```

```
// Response 1
{
  "data": {
    "person": {
      "firstName": "Luke"
    }
  },
  "hasNext": true
}

// Response 2
{
  "label": "HomeWorldDefer",
  "data": {
    "homeWorld": {
      "name": "Tatooine",
      “climate": "desert",
    }
  },
  "hasNext": true
}

// Response 3
{
  "label": "nameAndWorld",
  "data": {
    "firstName": "Luke",
    "lastName": "Skywalker",
    "homeWorld": {
      "name": "Tatooine"
    }
  },
  "hasNext": false
}
```

Example - old version

```
{
  me {
    id
    ... @defer(label: "1") { id bio } }
    ... @defer(label: "2") { age bio }
    ... @defer(label: "3") { dob bio }
    ... @defer(label: "4") { avatarUrl bio }
    ... @defer(label: "5") { name bio}
    ... @defer(label: "6") { joinedAt bio }
    ... @defer(label: "7") { lastSeen bio }
    ... @defer(label: "8") { location bio }
  }
}
```

Example - old version

Without @defer this is equivalent to

```
{
  me {
    id
    ... @defer(label: "1") { id bio } }
    ... @defer(label: "2") { age bio }
    ... @defer(label: "3") { dob bio }
    ... @defer(label: "4") { avatarUrl bio }
    ... @defer(label: "5") { name bio}
    ... @defer(label: "6") { joinedAt bio }
    ... @defer(label: "7") { lastSeen bio }
    ... @defer(label: "8") { location bio }
  }
}
```

```
{
  me {
    id
    name
    bio
    age
    dob
    avatarUrl
    joinedAt
    lastSeen
    location
  }
}
```

Example - old version

With @defer - response explosion

```
{
  me {
    id
    ... @defer(label: "1") { id bio } }
    ... @defer(label: "2") { age bio }
    ... @defer(label: "3") { dob bio }
    ... @defer(label: "4") { avatarUrl bio }
    ... @defer(label: "5") { name bio}
    ... @defer(label: "6") { joinedAt bio }
    ... @defer(label: "7") { lastSeen bio }
    ... @defer(label: "8") { location bio }
  }
}
```


```
{
  "data": {
    "me": {
      "id": 1
    }
  }
}
{
  "label": "1",
  "path": ["me"],
  "data": { "id": 1, "bio": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt
ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fug
mollit anim id est laborum."}
}
{
  "label":"2",
  "path":["me"],
  "data": { "age": 19, "bio": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididun
ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fug
mollit anim id est laborum."}
}
{
  "label":"3",
  "path":["me"],
  "data": { "dob": "19 BBY", "bio": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor inc
laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
deserunt mollit anim id est laborum."}
}
{
  "label":"4",
  "path":["me"],
  "data": { "avatarUrl": "....", "bio": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
deserunt mollit anim id est laborum."}
}
{
  "label":"5",
  "path":["me"],
  "data": { "name": "Luke", "bio": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor inc
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore e
mollit anim id est laborum."}
}
{
  "label":"6",
  "path":["me"],
  "data": { "joinedAt": "2024-09-10", "bio": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod t
ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
officia deserunt mollit anim id est laborum."}
}
{
  "label":"7",
  "path":["me"],
  "data": { "location": "XYZ", "bio": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor i
laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
deserunt mollit anim id est laborum."}
}
```

New Response Format

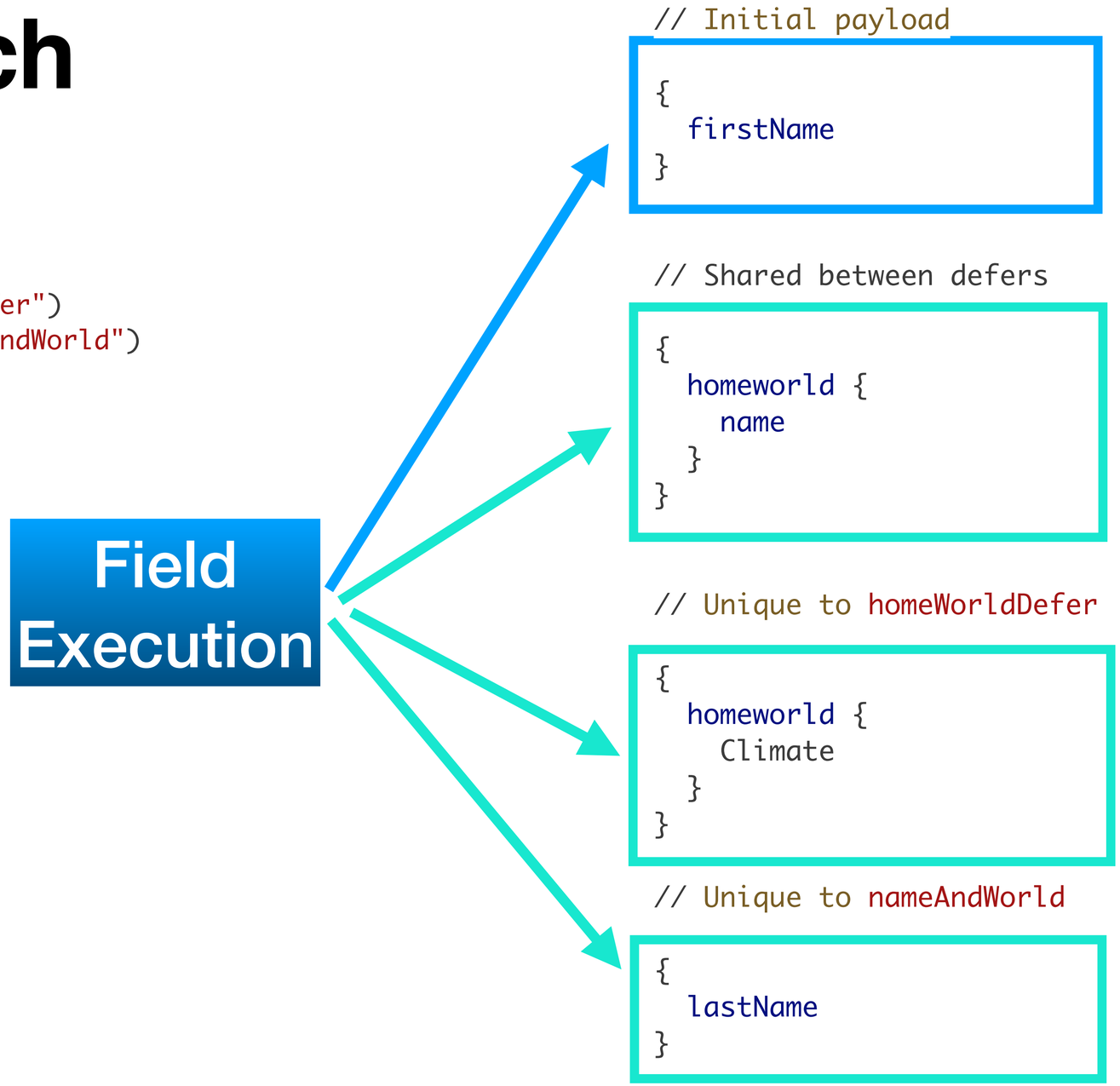
```
{
  me {
    id
    ... @defer(label: "1") { id bio }
    ... @defer(label: "2") { age bio }
    ... @defer(label: "3") { dob bio }
    ... @defer(label: "4") { avatarUrl bio }
    ... @defer(label: "5") { name bio}
    ... @defer(label: "6") { joinedAt bio }
    ... @defer(label: "7") { lastSeen bio }
    ... @defer(label: "8") { location bio }
  }
}
```


```
{
  "data": {
    "me": {
      "id": 1
    }
  },
  "pending": [
    {"id": "1", "label": "1", path: ["me"]},
    {"id": "2", "label": "2", path: ["me"]},
    {"id": "3", "label": "3", path: ["me"]},
    {"id": "4", "label": "4", path: ["me"]},
    {"id": "5", "label": "5", path: ["me"]},
    {"id": "6", "label": "6", path: ["me"]},
    {"id": "7", "label": "7", path: ["me"]},
  ],
  "hasNext": "true"
}
{
  "incremental": [
    {"id": 1, "data": {"bio": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehender
sunt in culpa qui officia deserunt mollit anim id est laborum."}}
  ],
  "completed": [{"id": 1}], "hasNext": "true"
}
{
  "incremental": [{"id": 2, "data": {"age": 19}}],
  "completed": [{"id": 2}], "hasNext": "true"
}
{
  "incremental": [{"id": 3, "data": {"dob": "19 BBY"}}],
  "completed": [{"id": 3}], "hasNext": "true"
}
{
  "incremental": [{"id": 4, "data": {"avatarUrl": "...."}}],
  "completed": [{"id": 4}], "hasNext": "true"
}
{
  "incremental": [{"id": 5, "data": {"name": "Luke"}}],
  "completed": [{"id": 5}], "hasNext": "true"
}
{
  "incremental": [{"id": 6, "data": {"joinedAt": "2024-09-10"}}],
  "completed": [{"id": 6}], "hasNext": "true"
}
{
  "incremental": [{"id": 7, "data": {"location": "XYZ"}}],
  "completed": [{"id": 7}], "hasNext": "false"
}
```

Execution approach

```
query {
  person(id: "1") {
    ...HomeWorldFragment @defer(label: "homeWorldDefer")
    ...NameAndHomeWorldFragment @defer(label: "nameAndWorld")
    firstName
  }
}
fragment HomeWorldFragment on Person {
  homeworld {
    name
    climate
  }
}

fragment NameAndHomeWorldFragment on Person {
  firstName
  lastName
  homeworld {
    name
  }
}
```



Execution approach

```
query {
  person(id: "1") {
    ...HomeWorldFragment @defer(label: "homeWorldDefer")
    ...NameAndHomeWorldFragment @defer(label: "nameAndWorld")
    firstName
  }
}
fragment HomeWorldFragment on Person {
  homeworld {
    name
    climate
  }
}

fragment NameAndHomeWorldFragment on Person {
  firstName
  lastName
  homeworld {
    name
  }
}
```

```
// Payload 1
{
  "data": {"person": {"firstName": "Luke"}},
  "pending": [
    {"id": "0", "path":["person"], "label": "HomeWorldDefer"},
    {"id": "1", "path":["person"], "label": "nameAndWorld"},
  ],
  "hasNext": true
}

// Payload 2
{
  "incremental": [
    {"id": 0, "data": {
      "homeWorld": {"name": "Tatooine"}
    },
    {"id": 0, "subPath": ["homeWorld"],
      "data": {“climate": "desert"}
    }
  ],
  "completed": [{"id": "0"}],
  "hasNext": true
}

// Payload 3
{
  "incremental": [
    {"id": 1, "data": {"lastName": "Skywalker"}}
  ],
  "completed": [{"id": "1"}],
  "hasNext": false
}
```

Actionable Payloads

```
query {
  person(id: "1") {
    ...HomeWorldFragment @defer(label: "homeWorldDefer")
    ...NameAndHomeWorldFragment @defer(label: "nameAndWorld")
    firstName
  }
}
fragment HomeWorldFragment on Person {
  homeworld {
    name
    Climate
  }
}

fragment NameAndHomeWorldFragment on Person {
  firstName
  lastName
  homeworld {
    name
  }
}
```

```
// Payload 1
{
  "data": {"person": {"firstName": "Luke"}},
  "pending": [
    {"id": "0", "label": "HomeWorldDefer"},
    {"id": "1", "label": "nameAndWorld"},
  ],
  "hasNext": true
}

// Payload 2
{
  // ❌ Not Actionable
  "incremental": [
    {"id": 0, "data": {
      "homeWorld": {"name": "Tatooine"}
    },
  ],
  "hasNext": true
}
```

Prevent Re-renders

```
query {
  person(id: "1") {
    ...HomeWorldFragment @defer(label: "homeWorldDefer")
    ...NameAndHomeWorldFragment @defer(label: "nameAndWorld")
    firstName
  }
}
fragment HomeWorldFragment on Person {
  homeworld {
    name
    climate
  }
}

fragment NameAndHomeWorldFragment on Person {
  firstName
  lastName
  homeworld {
    name
  }
}
```

```
// Payload 1
{
  "data": {"person": {"firstName": "Luke"}},
  "pending": [
    {"id": "0", "path":["person"], "label": "HomeWorldDefer"},
    {"id": "1", "path":["person"], "label": "nameAndWorld"},
  ],
  "hasNext": true
}

// Payload 2
{
  "incremental": [
    {"id": 0, "data": {"homeWorld": {"name": "Tatooine"}}},
    {"id": 0, "subPath": ["homeWorld"],
      "data": {“climate": "desert"}
    },
    {"id": 1, "data": {"lastName": "Skywalker"}}
  ],
  "completed": [
    {"id": "0"},
    {"id": "1"}
  ],
  "hasNext": false
}
```

Inline @defer or @stream

```
query {
  person(id: "1") {
    ...HomeWorldFragment @defer(label: "homeWorldDefer")
    ...NameAndHomeWorldFragment @defer(label: "nameAndWorld")
    firstName
  }
}
fragment HomeWorldFragment on Person {
  homeworld {
    name
    Climate
  }
}

fragment NameAndHomeWorldFragment on Person {
  firstName
  lastName
  homeworld {
    name
  }
}
```

```
// Payload 1
{
  "data": {
    "person": {
      "firstName": "Luke",
      "homeworld": {
        "name": "Tatooine",
        “climate": "desert"
      }
    }
  },
  "pending": [
    // No pending for homeWorldDefer
    // it was included in the initial data
    {"id": "0", "label": "nameAndWorld"},
  ],
  "hasNext": true
}
```

```
// Payload 2
{
  "incremental": [
    {"id": 0, "data": {"lastName": "Skywalker"}}
  ],
  "completed": [{"id": "0"}],
  "hasNext": false
}
```

Multiple @streams & batching

```
query {
  allFilms @stream(initialCount: 2, label: "filmsStream") {
    title
  }
  allPeople @stream(initialCount: 2, label: "peopleStream") {
    name
  }
}
```

```
// Payload 1
{
  "data": {
    "allFilms": [
      {"title": "A New Hope"},
      {"title": "The Empire Strikes Back"},
    ],
    "allPeople": [
      {"name": "Luke Skywalker"},
    ]
  },
  "pending": [
    {"id": "0", "path": ["allFilms"], "label": "filmsStream"},
    {"id": "1", "path": ["allPeople"], "label": "peopleStream"}
  ],
  "hasNext": true
}
```

```
// Payload 2
{
  "incremental": [
    {"id": "0", "items":[{"title": "Return of the Jedi"}]}
  ],
  "completed": [{"id": "0"}],
  "hasNext": true
}
```

```
// Payload 3
{
  "incremental": [
    {"id": "1", "items": [
      {"name": "C-3PO"},
      {"name": "R2-D2"},
    ]}
  ],
  "completed": [{"id": "1"}],
  "hasNext": false
}
```

Asynchronous list completion

```
query {
  allFilms @stream(initialCount: 2, label: "filmsStream") {
    title
  }
}
```

```
// Payload 1
{
  "data": {
    "allFilms": [
      {"title": "A New Hope"},
      {"title": "The Empire Strikes Back"},
    ]
  },
  "pending": [
    {"id": "0", "path": ["allFilms"], "label": "filmsStream"},
  ],
  "hasNext": true
}
```

```
// Payload 2
{
  "incremental": [
    {"id": "0", "items":[{"title": "Return of the Jedi"}]}
  ],
  "hasNext": true
}
```

```
// Payload 3
{
  "completed": [{"id": "0"}],
  "hasNext": false
}
```

Error handling

```
query {
  hero(id: "1") {
    ...HomeWorldFragment @defer(label: "homeWorldDefer")
    name
  }
}
fragment HomeWorldFragment on Person {
  homeworld {
    name # ERROR!!
  }
}
```

```
// Payload 1
{
  "data": {
    "hero": {
      "name": "Luke Skywalker"
    }
  },
  "pending": [{
    "id": "0",
    "path": ["hero"],
    "label": "homeWorldDefer",
  }],
  "hasNext": true
}

// Payload 2
{
  "incremental": [{
    "id": "0",
    "data": {
      "homeworld": {
        "name": null
      },
      "errors": [
        {
          "message": 'Oops!',
          "locations": [...],
          "path": ["hero", "homeworld", "name"],
        },
      ]
    },
  }],
  "completed": [{ "id": "0" }],
  "hasNext": false
}
```

Error bubbling

```
query {
  hero(id: "1") {
    ...HomeWorldFragment @defer(label: "homeWorldDefer")
    name
  }
}
fragment HomeWorldFragment on Person {
  # `homeworld` field is non-nullable in the schema
  homeworld {
    # `name` field is non-nullable in the schema
    name # ERROR!!
  }
}
```

```
// Payload 1
{
  "data": {
    "hero": {
      "name": "Luke Skywalker"
    }
  },
  "pending": [{
    "id": "0",
    "path": ["hero"],
    "label": "homeWorldDefer",
  }],
  "hasNext": true
}
```

```
// Payload 2
{
  "completed": [{
    "id": "0",
    "errors": [{
      {
        "message": "Oops!",
        "locations": [...],
        "path": ["hero", "homeworld", "name"]
      }
    }]
  }],
  "hasNext": false
}
```

Resources

- Discussions: github.com/graphql/defer-stream-wg/discussions
- Spec Edits: https://github.com/graphql/graphql-spec/pull/1077
- GraphQL-JS v17.0.0-beta.1
- **#incremental-delivery-wg** on the GraphQL Discord
- Demo Repo: https://github.com/robrichard/defer-relay-example

Rob Richard @rob_richard