GraphQL Metadata WG - July 2022

- No need for a separate WG repo, follow the InputUnions pattern
- At the InputUnion WG we had a bunch of competing solutions, to solve it we didn't 1-to-1 compare the solutions, instead we created a document with problem statement, solution criteria, and then compared each solution against these. This gave a single place for all the solutions and made comparison easier.
- Michael: I like the way the InputUnion pattern worked - it got a good picture of the problem: what are we trying to solve, and what are the downsides of each solution?
- Roman: input unions is nice to have, but metadata is a big deficiency and we urgently need to solve it - pick a solution and go. It's a big surprise to people that it doesn't work this way. Need to explore and analyze several approaches and then pick which one is better.
- Ivan: Core idea is to have common problem statement and solution criterias. We need to agree what we're solving.
- Michael: I know this is missing, but I wouldn't say it's that urgent - there are workarounds, but the solutions we have currently have downsides - when we put it in the spec we shouldn't rush it.
- Ivan: something can become the defacto specification.
- Michael: three big implementations have it already - all of them it's an opt-in, so it's not a defacto standard.
- Roman: But if we put this in the spec then it's not a workaround. That you've implemented it proves it works, right?
- Michael: no it doesn't work; like with language designers you try things out and sometimes you see it's not good and you pull it. I've seen issues like double serialization - it's not nice to use and has problems that you discover at runtime, whereas we could have a fully static solution. By workarounds I meant e.g. Apollo Federation with the SDL. There are different workarounds for each framework. It's an experiment to see if it works out, but I think it doesn't. But we should discuss it. Would be good to see the metrics of what does/does not work with each approach.
- Roman: there's a problem with returning input types/etc. This is a problem with any solution. This should become a separate issue in general - there is a need for this.
- Michael: that's one of the reasons we have the "as" - this input is represented as that output. Benjie's working on an alternative - struct - it's more general, but it also has downsides and plussides. It's important to look at that precisely.
- Ivan: I'm proposing criteria that we judge the final solution. Assume that we can add anything (struct, etc). Struct as a solution for metadata requires you use struct, introspection is one of the use cases for struct. So instead of focussing on it, let's assume we can add anything to the spec. So judge the solutions based on the underlying features being in the spec (e.g. we can assume that "struct" is already supported). If struct is useful beyond introspection, it should be added independently of introspection.

- Ivan: secondly; let's focus on goals rather than method. We want typed value - discuss and agree it's a requirement. Then discuss how the different approaches solve this. Isomorphic types / "as" / some other solution. I've drafted a document, but I think it would be fair and more productive if you give your solution criterias.
- Michael: share something you already have.
- https://github.com/graphql/graphql-wg/blob/ab4279ab095cfd6a8c9f60b1be8e52578f6c7f34/rfcs/CustomMetadataOverIntrospection.md
- Benjie: With input unions we introduce weights to the criterias. They can also be controversial.
  We first collected all the criteria we thought of and then went over them and dropped the ones we did not want in the end.
- <Copied rules into bottom of these notes, editable by team>
- Roman: is it metadata? We're adding pieces of custom data, not metadata. Data elements. Label. Hey, hello from me. Anything. I'm afraid that calling it "metadata" skews our thinking. No; it's random crap. Allows people to make mini contracts and use that.
- Ivan: schema is exposed by server. Metadata is "data about data".
- Michael: it's annotations. Let's not get stuck on the term.
- Roman: so long as we understand this and agree, let's continue.
- Ivan: I'm careful not to use the word "directive" because I don't want to shape the solution.
- Michael: let's just collect them first and then we can go over them later.

- A. It should be possible subselect metadata values
  Following general GraphQL principle clients should ask only data that they are interested in. Note: it should work both for top-level metadata values and also different fields inside this value. Good example of why it needed is a top-level "translation" value that can includes dozens of sub-fields one for each language.
    - Roman: I don't think this is needed
    - Michael: In C# you don't need that because the object is in memory, but in GraphQL the data is remote, so it has different needs.
    - Michael, Benjie: there can be a lot of metadata quite quickly
    - Benjie: I think subselection is critical. We don't just use introspection for full schema description, but also for specific queries e.g. to populate dropdowns.
    - Ivan: OpenAPI is an example; people put a _lot_ of client-specific things there. For example generating SDKs based on OpenAPI definition meant that people added a lot of metadata. I expect we'll face a lot of the same problems: people adding a lot of metadata.
    - Roman: I hope the nightmare of REST/OpenAPI won't repeat itself here in GraphQL.
    - Roman: reasonable people once they have two languages would find a different solution. One typo would require rebuilding everything.
    - Roman: don't assume that devs will do unreasonable things even if we give them the tools to do so.

    - Michael: distributed schemas already use a lot of metadata.

- B. It should be possible to dump all metadata
  Some clients (e.g. GraphQL proxies, schema stores, etc.) should have ability to dump all data required to generate exactly the same introspection result as original server.
    - Metadata must be static - cannot include random numbers, current date, etc.

- C. It should be possible to deprecate specific metadata value using existing deprecation mechanism
  Note: It should be possible to deprecate both "metadata usage" and "metadata definition" separately.
    - i.e. when writing the schema you can still add the deprecated metadata to new places without warning, but when _consuming_ the schema you should be notified the metadata you request is deprecated

- D. Metadata value should be typed and introspectable
- E. It should be possible to attach semantic meaning to a "metadata" using existing "specifyByURL" mechanism
- F. Metadata values should have descriptions as part of introspection
- G. It should be possible to attach metadata to metadata definitions
    - Schema composition - metadata may be intended for schema composer. We shouldn't expose it further (implementation detail).
    - This is a real issue for federation.
- H. Implementation details shouldn't be exposed as metadata
    - It's not about secrets; we should not be exposing all the crap we have on the server - if something was used during schema building time/server initialization and it's explicitly not meant to be exposed, we shouldn't expose it.

- Benjie: Note I added some of these as anti-goals, discussion points so we can figure out what are the non-goals as well as what are the goals.

- I: It should be representable by SDL
- J: Must support polymorphic data
- K: Must support large (multi-megabyte) data
- L: Metadata must be static
    - Same response should be generated from same introspection query for the same instance of a schema.
    - Data may still change when schema is updated (new version of the schema).
- M: It should be clear if changing metadata will break clients.
- N: It should allow for "any" data type.
    - (Reject)
- O: It should allow for dynamic data.
    - (Reject)
- P: It should allow arguments (like `__type(name: String!)`)

    - Even if the data is still static (same arguments, same response)
    - Ivan: Introspection query doesn't use `__type`, it just uses `__schema` and gets all the data that way. Arguments indicating which subfields to return could be acceptable.
    - Michael: we can do similar with `isDeprecated`
- Q: Should be familiar with known concepts
    - Roman: like directives.
    - Michael: we should add known solutions to the document
    - Anthony: we need to outline what those concepts are once we've proposed some solutions. Making it a blanket statement could be overly broad, and may or may not apply to any existing solution.
- R: It should avoid confusion with directives
-
-
-
- Readability was discussed - you should be able to read the SDL with metadata fairly easily.
-
-
-
- Anthony: we need a better story about _why_ this is useful. I know Apollo Federation is a use case, but if that's the only use-case, why is it important? We should make it clear how things are valuable to different people.
- Michael: to be clear, all the different distributed schemas need metadata, not just Apollo.
- Anthony: if distributed schemas is not the only use case, we should document the other use cases.
- Benjie: I have a lot of use cases; I've wanted this for ages for PostGraphile (not distributed schemas).
- Ivan: @experimental is another use case.
- Ivan: Action points
- Date: we'll ask in thread
- Roman: let's just be active on the threads of discussion. Let's agree.
- Michael: we had separate documents for the different proposals for InputUnions in the end. Discussing competing solutions in the same thread makes it very confusing.
- ACTION - I'll aggregate things from the document and we'll update the PR from draft and ping you all and we can continue to work on it.
-