JSONPath

September 17th (Friday), 09:00-11:00 UTC
(11:00–13:00 CEST, 02:00–04:00 PDT)

From a 2021-06-15 slide

What is our stance on implicit conversions?
(Emerging consensus was: No implicit conversions.)

What about conversion to Boolean ("truthy")?

Background: Result type of JSONPath query

Within a selector chain, each selector returns a nodelist.
This nodelist is then input to the next selector.

Using queries in the expression language:
How are nodelists used/translated into the terms of the expression language?

`exist-expr` **vs.** `comp-expr`

@-based queries limited to `@.foo/@[expr]`
In filter expressions, this is a single node or no node

Nodelist implicit conversion:
Can easily translate to Boolean (existence) for `exist-expr`
Can translate to union of JSON value and "undefined" for comp-expr

Current grammar also allows more general JSONPath queries can result in nodelists with more than one entry

Datatypes in expression language

`comp-expr` currently compares against

- "primitive" JSON types, or
- query results ("nodelists")

What about `$..foo == @.foo`

No place for structured values (not even literals).

No computation (compare against literals[1] only).

[1] or query results

(2021-06-15) Example: Comparison with structured values

Should comparison with structured values (e.g., @.foo == [1, 2]) be supported?

If it is not supported, should this silently fail or the attempt cause a syntax error (in #99, it causes a syntax error, but then the text says something else).

— Data types: can we even write and pass around `[1, 2]`?

— Should comparison with structured values be allowed?

```
* Comparisons are restricted to primitive values `number`, `string`, `true`, `false`, `null`.
  Comparisons with complex values will fail, i.e. no selection occurs.
```

(cabo:
This first requires defining literal and/or constructor syntax for structured values.)

An expression should be able to operate on any JSON literal. I see no reason why @.foo == [1, 2] should be disallowed.
— Greg

What we probably need to decide first

(1) type space in the expression language:
• limited subset of JSON + query results
* full JSON + query results

(2) extent of computation
(2a) extent of implicit conversion in those computations

(3) extent of query integration (`@..foo`?)