JSONPath

November 10 (Wednesday), 12:00-14:00 UTC
(13:00–15:00 CET, 04:00–06:00 PST)

#127

Should we provide for structured types in containable? #127

Sub-issues:

— value notation for structured types

— comparison semantics for structured types

value notation for structured types...

```
containable  = path /                          ; path to primitive value
               number /
               string-literal
container    = path /                          ; resolves to array
               array-literal
```

; array-literal UNDEFINED
cf.:

```
comparable   = number / string-literal /       ; primitive ...
               true / false / null /           ; values only
               path                            ; path value
```

...value notation for structured types

Cannot just copy JSON, as we have different syntax ('a'...)

Allow `path` in array/object constructor?
(Proposal: if we do this at all, yes.)

comparison semantics for structured types

Would need to write up the obvious.

#127 Resolution?

#120 says "(From 2012-09-17 interim:)
So we would have syntax for array literals composed of primitive values only (subset of JSON array literals);
other JSON array literals cause a syntax error."

Did we change our mind?

#124

JsonInclude Processing Extension #124

Proposal: Not in base RFC.

#123

Is absent equal to absent? #123

Proposal: empty nodelist converts to "undefined" in comparison (i.e., yes!)

Underlying assumption: The domain/range of expression language constructs is not limited to JSON values

#122

Nodeslists on RHS of in-op #122

```
@.color in $..allowed_color
```

Assumption: The semantics of "in" is based on that for "==", just scanning the RHS array.
Of course there is the more general problem what a nodelist with more than one entry means for comparison:

```
@.color == $..allowed_color
```

#120

regex-expr and contain-expr are missing lhs #120
(they no longer are)

```
regex-expr   = (path / string-literal) S regex-op S regex
contain-expr = containable in-op container
containable  = path / ; path to primitive value
               number / string-literal
```

Note that this is missing false/null/true -- OK?

#119

Semantics of @.foo as a standalone expression (truthiness) #119

Proposal: In a boolean context (filter expression, !, &&/||), a path converts to false if nodelist empty, true otherwise

(1) we have no other existence test
(2) gets rid of "truthiness" weirdnesses

#118

key missing vs key present with null #118

This is the same thing again.
Confusion: Direct conversion to Boolean (existence of node) vs. comparison (extracts value from node)

#117

Normalization #117 (Unicode)

read https://www.w3.org/TR/charmod-norm/

`https://cburgmer.github.io/json-path-comparison/results/filter_expression_with_equals_string_in_NFC.html (Mot\u00f6rhead and Moto\u0308rhead), consensus seems to be to not normalize.`

Proposal: Byte string comparison (~ code point sequences)
Could add normalizing comparison, indexing[!]?

#109

filtering w/o child selection #109

#103 (closed)

Partial determinism of the descendant selector (..) #103

Proposal: stick with
B. Require `..` to visit arrays in array order and visit nodes before their descendants
(Obviously, "objects" are unordered in JSON, so there is the non-determinism)
➔ PR #134 (merged)
Continue at #23

(See also #27, #60)

#95

ER - Provide syntax for returning a specific number of filtered elements #95

Proposal: Keep pagination in the APIs to JSONPath, not in the actual JSONPath expressions

#93

"Index-selection query" in bracket notation #93

Not sure we have a proposal yet.

#92

bracket notation #92

(No idea if there is anything left from this discussion)
Proposal: close.

#78

What do we want our RFC to do? #78

Discussion item.
Proposal: close.

#70

Regular expressions in filters #70
Already decided: RE literals only, no compute.

— Select (define) one regular expression flavor
— Provide a way to plug in regular expressions
— Not in base RFC (but keep an extension point)

select one regular expression flavor

No implementation consensus.
So we get to choose.
Principle of least surprise vs. interoperability

Parsing/searching REs vs. matching REs

— Select a version of ECMAScript (parsing/searching RE)
— Select W3C XSD RE (matching RE)
— Build "modest subset" (e.g., iregexp)

#69

Examples Appendix #69

Yeah, should make one, when we have time...

#67

Differentiation from JSON Pointer #67

Write a passage contrasting JSONPath to JSON Pointer.
Placeholder already in Intro.
Need to write text, with suggestions from Mark Nottingham and Carsten Bormann
(See also #44)

#64

Filter Expressions #64

This is another issue that rehashes just about everything.
Filter expressions work on arrays and objects.
Editorial work needed.

#63

Respect Implementations #63

Discussion item.
Proposal: close.

#62

Error Handling #62

Discussion item; records consensus that there are no runtime errors (just failed matches), i.e., the error behavior of a JSONPath expression is data-independent.
Implemented in Section 3.1.
Proposal: Close.

#61

Array Slice Operator #61

Originally raised [::0], which is now decided in 3.5.6. Proponents of this issue should read that closely and close the issue unless there is one.

#59

Relative paths support #59

OBE -- can we close?

#58

A "General Considerations" section #58

Dump of text from json-schema.org drafts; mostly OBE.
Contains a definition of a "modest subset" of ECMAScript REs (which requires the insertion of anchors), which probably needs some fixing.
Proposal: close.

#55

WoT discovery, geolocation, and JSON pointers #55

Actual user requirements, for once.
(With some confusion about where JSON pointers go in a URI, clarified).
Proposal: get updated info; make sure we meet these requirements.

#53

Hello Issue #53

Discussion item, now mostly OBE.
Proposal: close.

#49

Proposal - Selecting a subset of an object's properties #49

Interesting syntax proposal for
~ relational algebra projection as in SELECT x, y...

Resolution was: "jsonpath selects and doesn't transform"
Proposal: close

#44

Including value locations in output #44

Turned into an interesting discussion of JSON Pointer, which should be reflected in the resolution of #67

#37

Scrap or apply RFC 2119 language #37

Resolution:
Like soy sauce, BCP 14 (RFC2119+RFC8174) language is an essential addition, but only works well if applied sparingly. [...] a continuing task during editing of this document.

Proposal: Close (but keep in mind).

#25

Security items #25

Discussion item.

Proposal: Write the Security Considerations section, close this.

#23

Duplicates in selector output #23

Discussion confused by the lack of distinction between duplicate values and duplicate nodes.
Pretty clear that we don't remove duplicate values.
PR #134 makes it less attractive to remove duplicate nodes.
Proposal: Don't.

#21

"Union" could have more description, and maybe a new name #21

Discussion item without much discussion.

Proposal: close.

#88, #59, #17, #15, #14

label:"overtaken by events -- can we close this now?"

Please check, and then:
Proposal: close.

Unions revisited #88
Relative paths support #59
Query expression language support #17
Selector definitions (basically an analysis and description of the DSL) #15
Define an algorithm #14