RJP: Why?

- Data location is known relative to an *unknown starting point*
  - JSON Schema annotations can incorporate instance data, but can...
    - match multiple locations, e.g. every array element
    - be re-used in unanticipated contexts
- URI fragments are opaque in terms of relative URI-reference resolution
- Avoid the complexity of queries or wildcard / regex matching
  - Relative JSON Pointer simpler and faster
  - Intended for use where only a single match is correct, based on relative location
RJP: What?

relative-json-pointer = origin-specification ( index-or-name-of / json-pointer )
origin-specification = non-negative-parent-count [ index-adjustment ]
index-adjustment = ( "+" | "-" ) positive-index-count
index-or-name-of = "#"

- Distinct from JSON Pointer, which is always either empty or starts with "/"
- Examples:
  - "0" – value at starting point (with a JSON Pointer component of "")
  - "1#" – name or index of parent with respect to grandparent
  - "0+1/foo" – value of "foo" property of next array item after the current location
RJP: How?

- Current specification: Evaluate with a document and starting location
  - Challenging because parent / sibling access is not part of the JSON data model
  - At least one JSON Pointer library declined to support RJP because of this

- Proposal: Resolve against a base JSON Pointer and apply to a document
  - Reduces implementation to trivial steps plus existing JSON Pointer implementation
  - Could potentially allow non-error-checked partial resolution…
    - …to a regular JSON Pointer
    - …to a name or index (but ambiguous as to which)
    - …to an unresolved "take index of one past the last array item" operation
  - Expands usage options and will likely enable broader implementation support
RJP: Where?

- Known implementations supporting **index adjustment** (Dec. 2020 draft on)
  - **JavaScript/TypeScript**: @fosfad/relative-json-pointer
  - **Python**: jschon JSON Schema & other JSON utilities package
  - **Go**: 0x51-dev/jsonptr (new as of this month, possibly unpublished)

- Known implementations of earlier drafts (with all other features):
  - **JavaScript/Typescript**: several, notably json-ptr (~500K weekly downloads; not just RJP)
  - **Perl**: JSON::Pointer
  - **PHP**: Opis JSON Schema suite
  - multi-language including **.NET**: JSON Essentials™ for COM/ActiveX
Example: OpenAPI Specification document parsing

```json
1 {
2     "title": "OAS 3.0 Schema Object",
3     "type": "object",
4     "properties": { ... },
5     "oasExamples": {
6         "examples": [
7             "0/example",
8             "0/default"
9         ],
10        "schemas": [
11            "0"
12        ]
13     }
14 }
```
Example: JSON Hyper-Schema links (in YAML)

Instance:

1 - authorId: 1
2 books:
3 - bookId: 101
4 - bookId: 102
5 - authorId: 2
6 books:
7 - bookId: 200

Hyper-Schema (type: array / object omitted for space):

1 items:
2   properties:
3   authorId: { type: integer }
4   books:
5     items:
6       properties: { bookId: { type: integer } }
7       links:
8         - rel: author
9         href: /people/{personId}  # resolve id from grandparent
10     templatePointers: { personId: "2/authorId" }
11       - rel: publication
12       anchorPointer: "2"  # attach link to grandparent
13       href: /books/{bookId}  # resolve bookId locally
14         - rel: next
15         href: /books/{bookId}  # resolve bookId from next item
16     templatePointers: { bookId: "0+1/bookId" }