Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 5, 2015

D. Oskarsson
JSOND.org

March 9, 2015

JavaScript Object Notation Definition (JSOND)

draft-oskarsson-jsond-00

Abstract

JSOND (JSON Definition) is a simple, yet powerful, definition language for JSON text.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 5, 2015.

Copyright Notice

1. Introduction

JSOND (JSON Definition) is a simple, yet powerful, definition language for JSON text.

The purpose of JSOND is to facilitate development and documentation of JSON text.

JSOND is designed to be a minimal superset of JSON.

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. JSOND Grammar

JSOND text is JSON text that MAY include JSOND grammar. JSOND grammar is a superset of JSON grammar [RFC7159] [RFC4627] [ECMA-404]. The rest of this document describes the JSOND grammar in ABNF [RFC5234] and text.

2.1. Values

A JSOND value MUST be an object, array or a string.

```
value = object / array / string
```

2.2. Objects

A JSOND object MUST define all members in the corresponding JSON object. A JSON object MUST NOT contain a member that has not been defined in the corresponding JSOND object.

```
value = object                          ; e.g. { "a": "string" }
```

2.3. Arrays

A JSOND array defines zero or more values. All values in a JSON array MUST be defined by at least one of the values in the corresponding JSOND array.

```
value = array                           ; e.g. [ "string" ]
```

2.4. Booleans

A JSOND boolean defines that the corresponding JSON value MUST be true or false.

```
value = %x22.62.6f.6f.6c.65.61.6e.22    ; "boolean"
```

2.5. Strings

A JSOND string defines that the corresponding JSON value MUST be any string.

```
value = %x22.73.74.72.69.6e.67.22        ; "string"
```

Regular expressions [ECMA-262] MAY be used to define a subset of strings.

```
value = %x22 regular-expression %x22     ; e.g. "[a-z]"
```

2.4. Numbers and Integers

A JSOND number defines that the corresponding JSON value MUST be any number.

```
value = %x22.6e.75.6d.62.65.72.22        ; "number"
```

A JSOND integer defines that the corresponding JSON value MUST be any integer.

```
value = %x22.69.6e.74.65.67.65.72.22     ; "integer"
```

2.4.1 Sets and Intervals

An arbitrary number of mathematical sets and intervals [ISO-80000-2] MAY be used to define a subset of numbers.

A corresponding JSON number MUST match a number in the defined subset.

```
begin-exclusive = %x28                   ; (

end-exclusive = %x29                     ; )

integer = [ minus ] zero / ( digit1-9 *DIGIT )

number = integer [ frac ] [ exp ]

set = begin-object number *( value-separator number ) end-object

interval = begin-array / begin-exclusive ( ( number
value-separator ) / ( number value-separator number ) /
( value-separator number ) ) end-array / end-exclusive

value = %x22 1*( set / interval ) %x22   ; e.g. "[1.0,2.0)"
```

Set elements SHOULD be ordered in increasing order from the least to the greatest element. There must be at least one element.

The left or right interval endpoint is OPTIONAL. An undefined left endpoint defines negative infinity. An undefined right endpoint defines positive infinity. If both endpoints are provided the left endpoint MUST be less than the right endpoint.

An interval that is declared using integers defines the corresponding subset of integers. An interval MUST be declared using at least one number with an explicit decimal component to define a subset of real numbers. The decimal component MAY be .0.

Insignificant whitespace is OPTIONAL in sets and intervals.

2.5. References

Any JSOND value MAY be persisted as a file. A file SHOULD be referenced using a relative path, an absolute path, or using the http or https scheme [RFC3986].

```
value = %x22 [ scheme ] path %x22           ; e.g. "file.jsond"
```

JSOND files SHOULD have the filename extension .jsond. Circular references SHOULD be avoided.

2.6. Constants

A value that is not valid JSOND grammar SHOULD be interpreted as a constant and thus REQUIRED in the corresponding JSON text.

The literals, true, false, and null are not valid JSOND grammar. Numbers are not valid JSOND grammar.

Most strings are valid regular expressions and thus valid JSOND grammar. String constants SHOULD include boundary matchers.

2.7. Optionals

A member can be defined as optional by appending a question mark to the end of the name.

```
name = %x22 *char [ %x3f ] %x22          ; e.g. "name?"
```

An optional member MAY have the value null. An optional member MAY be undefined in JSON text.

3. IANA Considerations

Additional information:
Magic number(s): n/a
File extension(s): .jsond
Macintosh file type code(s): TEXT

4. Security Considerations

See Security Considerations in [RFC7159] Section 12.

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3986] Berners-Lee, T., Fielding R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

5.2. Informative References

[ECMA-404] Ecma International, "The JSON Data Interchange Format", Standard ECMA-404, October 2013, <http://www.ecma-international.org/publications/standards/Ecma-404.htm>.

[ECMA-262] Ecma International, "ECMAScript Language Specification", Standard ECMA-262, June 2011, <http://www.ecma-international.org/publications/standards/Ecma-262.htm>.

[ISO-80000-2] International Organization for Standardization, "Quantities and units - Part 2: Mathematical signs and symbols to be used in the natural sciences and technology", Standard ISO 80000-2:2009, December 2009, <http://www.iso.org/iso/home/store/catalogue_tc/catalogue_tc_browse.htm?commid=46202>.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

Appendix A. Examples

Example 1:

```
[
   {
      "id": "integer",
      "slug": "string",
      "url": "string"
      "category": "integer",
      "price": "number",
      "reduced": "boolean",
   }
]
```

Example 2:

```
[
   {
      "id": "[0,)",
      "slug": "[a-z0-9]",
      "url": "url.jsond"
      "category": "{10,25,50}",
      "price": "(0.0,)",
      "reduced?": "boolean",
      "margin": "(high|medium|low)",
      "available": true,
   }
]
```

Author's Address

Daniel Oskarsson
Satuna Vekabacken
SE-541 94
SWEDEN

Email: daniel@jsond.org