Introduction

cswr.github.io/JsonSchema/spec/introduction

The basic concepts needed before diving into the world of JSON Schema
Here we present a brief description of some concepts that will make the reading of the specification easier. The first section explains what a JSON document is, which is essential for understanding the definition of JSON Schemas and how they work. The second section contains a simple explanation of what a JSON Schema is together with a small example. Finally, in the third section we present some notation that is used throughout the specification to avoid ambiguities.

JSON

JSON (JavaScript Object Notation) is a file format commonly used for sharing information over the Web. JSON files are lightweight and easy to ready both by machines and developers. This has made JSON the most popular file format for client-server communication. A JSON instance is one of the next seven elements:

- `string` A string is a sequence of valid characters between quotes. For example, `"Hello!"` is a string, but `23` isn't.
- `number` A number is created using JSON notation. For example `-2`, `23`, `3.14` and `2.54e3` are all numbers in JSON.
- `integer` An integers is a JSON number without a fractional part. For example, `34` and `-91` are integers.
- `boolean` A boolean JSON object is one of two values: `true` or `false`.
- `null` A null object in JSON is the value `null`.
- `array` A JSON array is an ordered sequence of JSON documents separated by commas and enclosed in brackets. For example, `["Hello", 23, true, null]` is a JSON array, as well as `[-2, -1, 3.14, 1.62e10]`.
- `object` A JSON object is a set of pairs of the form `key:value`, where `key` is a JSON string and `value` is a JSON document. These pairs are separated by commas and enclosed in braces. JSON objects are the most common type of JSON documents. For instance, to describe specific information about a product, we might want to remember its *id*, *name* and the quantity of the product remaining in stock. The following JSON object contains this information:

milk.json

```
{
     "id": 1,
     "name": "Milk",
     "stock": 25
}
```

Basic JSON constructs described above can be combined to create more complex documents. To illustrate this consider a situation where we want to store information about multiple products. In order to do so we could use an `array` of `objects` of the form above. As an example consider the following JSON document.

products.json

```
[
    {"id": 1,
     "name": "Milk",
     "stock": 25},
    {"id": 2,
     "name": "Yogurt",
     "stock": 15,
     "link": "http://delicious-yogurt.com"}
]
```

Here our JSON document is an `array` that consists of two `objects`, the first one storing information about Milk and the second one about Yogurt.

JSON Schema

A JSON Schema is a file that specifies the structure of JSON documents that are used in a certain application. For example, owners of a store might want to share documents with the information about products. Here, a JSON Schema could specify that each such document is a list of products, and that each product contains an *id*, a *name*, a *stock* and optionally a *link* to the manufacturer's website. Since it is very simple to verify if a JSON document *satisfies* a schema, using a schema would be useful to ensure that all shared documents have this structure. The next JSON Schema specifies the structure of an `array` **products.json** defined above:

product_list_schema.json

```
{
    "type": "array",
    "items": {
        "type": "object",
        "properties": {
            "id": {"type": "integer"},
            "name": {"type": "string"},
            "stock": {"type": "integer"},
            "link": {"type": "string"},
        },
        "required": ["id", "name", "stock"],
        "additionalProperties": false
    }
}
```

First, `"type": "array"` indicates that the document must be an array. Then, `"items": {...}` tells us that each of the array's items must satisfy what's specified in the braces (which is also a schema). Hence, `"type": "object"` indicates that each item of the array must be an object, and `"properties": {...}` indicates what should be satisfied (also a schema) by the properties of each product. Finally, `"required": ["id", "name", "stock"]` specifies that each product must have an *id*, a *name* and a *stock*, and `"additionalProperties": false` says that a product cannot have properties that are not mentioned in `"properties": {...}` (and hence the only optional property is *link*).

Note that the file **products.json** is *correct* according to what is specified in **product_list_schema.json**.

Notation

JSON Document: A document containing a JSON instance in its content.

JSON Schema Document: A document containing a JSON Schema instance in addition to a set of JSON Schema instance definitions in its content.

JSON Schema: A JSON Schema instance. Note that these kind of instances do not include definitions on their structure.

definitions : A set of auxiliary schemas whose purpouse is to be reused and combined later on the schema or in other definitions. You can take a look at the precise formalization here.

key-value pair: A key-value pair represents an assigment from a particular keyword to a JSON instance.

restriction: A single condition that a JSON instance must satisfy. A JSON Schema can be seen as a set of restrictions.

We use a simple, visual-based Extended Backus-Naur Form (EBNF) notation to specify how documents are written. You can look at the Precise Definition.

Where to go from here

You can visit our User Guide for a quick reference on how to create JSON Schemas. If you want to understand in detail how a keyword is validated, please go to the corresponding section of the specification. The specification is intended to be exhaustive and should contain the information needed to write your own JSON Schema validator (if this is not the case please open an issue).