javaScript和JSON (进阶)

> ! 在javaScript和JSON这篇博文中已经对JSON的基础知识进行了系统的介绍，此文是JSON知识的**进阶内容**，本文输出和JSON有关的以下内容：

> ❒ JSON核心
> ❒ JSON的校验(JSON Schema)

JSON核心

JSON是一种数据交换格式，是当前网络通信中使用的主流数据格式。

JSON本身并不局限(依赖)于某项特定的技术，非私有且可移植，几乎所有的现代编程语言(`javaScript` | `java` | `Ruby` | `C#` | `PHP` | `Object-C` 等)和平台都支持对JSON数据的序列化和反序列化处理，JSON主要应用在 网络通信的数据格式 、 Node用来存储项目元数据 、 Kafka类似的消息平台 以及 MongoDB等NoSQL数据库 中。

JSON流行的主要原因

- **JavaScript语言的复兴和崛起。**
- **JSON自身数据结构的简洁和紧凑特性。**
- **基于JSON的RESTful API呈现大规模增长。**
- **Ecma国际和IETF相关的标准化工作让JSON获得行业认可。**

JSON的作者Douglas Crockford在创作时借鉴了JavaScript对象字面量的语法，也就是说JSON本身就是JavaScript对象字面量表示法的一个子集和JavaScript开发能够无缝融合，而JavaScript编程语言的复兴和崛起(前端 + Node后端生态)极大的推动了JSON的流行。

JSON的数据表示方式非常简洁，结构紧凑易于阅读而且其本身的结构与高级编程语言中的 对象|字典|数组 结构天然一致，与XML相比更适合面向对象的设计和开发。此外，JSON格式的文档通常比相同内容的XML文档更小，因此在进行网络传输和处理的时候更快、效率更高，JSON本身的这些特性加上相关技术环境的发展让它逐步替代XML成为互联网中主要的数据交换格式。

近些年，基于JSON的RESTful API呈现大规模爆炸式增长，包括LinkedIn、Github、Twitter、Facebook、Tumblr和Amazon等公司都提供基于JSON的RESTful API(备注 相关API可以访问programmableweb查询)。

JSON的标准化(成为一项技术标准)，让JSON获得了行业内的认可，下面简单列出主要的标准化进程。

```
2001 年 JSON由Douglas Crockford提出。
2006 年 JSON由IETF通过RFC 4627进行首次标准化。
2013 年 Ecma国际通过ECMA 404 将JSON正式标准化。
2014 年 Tim Bray发布了RFC 7158和RFC 7159作为原始标准的改进版本(主要修正了4627标准中的一些错误)。
```

JSON数值的类型

JSON数值的类型主要指的是在JSON文档中，键值对冒号(:)后侧值的数据类型，主要包括：

- **null**
- **数值**
- **对象**
- **数组**
- **字符串**
- **布尔值**

`null` 是JSON中的一个特殊值，用来表示某个key(属性)没有值用作占位符，注意不能由引号括起来。

数值 遵循JavaScript双精度浮点数格式，支持指数形式，但仅支持十进制数不支持8进制和16进制数。

对象 由 { 和 } 把键值对(key-value)括起来，允许设置为空对象，可以内嵌在其他的对象或者是数组中。

数组 由 [和] 把元素括起来，允许设置为空数组且不限制类型，可以内嵌在其他的对象或者是数组中。

字符串 由包含在双引号间的N(>=0)个Unicode字符组成可包含由转义字符，但单引号字符串是非法的。

布尔值 只存在true和false这两种值，且不能用引号把它们括起来，需注意JSON中没有为**undefined**值。

More ...

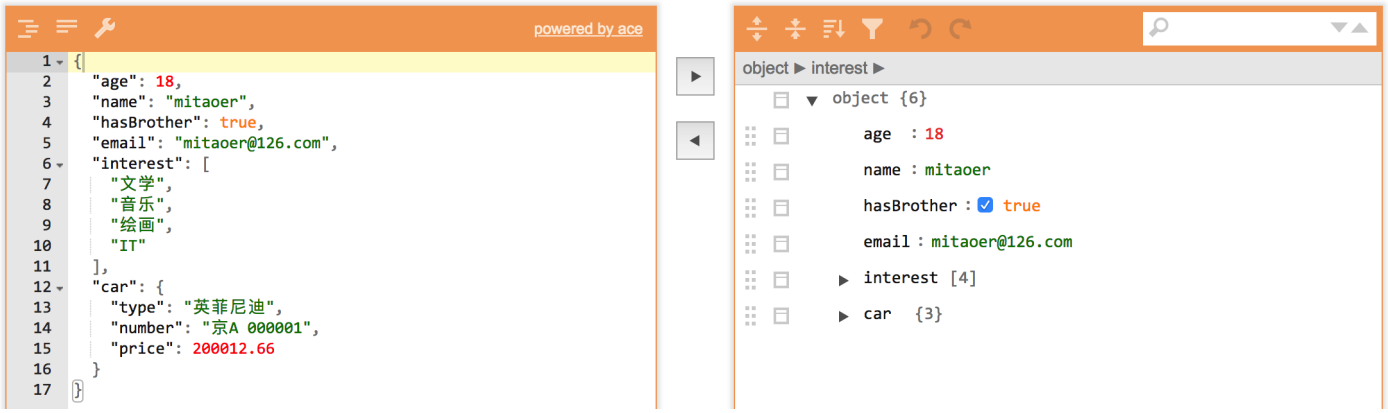
版本➤JSON的核心标准不会再有新的版本，适用”无版本”理念。

类型➤JSON数据在文件系统中存储的标准文件类型为.json，IANA为JSON文档指定的MIME(媒体类型)为application/json。

缩进➤JSON的编码规范并不存在与JSON数据缩进相关的话题，主要因为JSON本身是一种序列化格式而非呈现格式，所以缩进对JSON本身而言意义不大，在优化JSON现实的时候，常见的缩进方案是两格缩进或四格缩进。

工具➤

在线生成合法JSON文档的工具(1) JSONmate
在线生成合法JSON文档的工具(2) JSON Editor Online
在线校验JSON文档是否合法的工具(1) JSON Schema
在线校验JSON文档是否合法的工具(2) JSON Validate
快速生成大量测试JSON数据的工具推荐 JSON Generator
在线对复杂JSON数据格式化的工具推荐 JSON 在线格式化



! 创建|编辑JSON文件的时候，建议使用JSON Editor Online在线工具，上图是其基本工作界面。

JSON Schema

JSON Schema是对JSON文档中的内容、结构和格式进行的声明，用于校验JSON文档。区别于普通的JSON校验工具，JSON Schema能够对JSON文档执行**语法校验**和严格的**语义校验**。JSON Schema这种能够用于校验JSON文档内容和语义的工具能够有效提供服务的安全性，在消息系统中的应用能够确保数据格式的正确性，在API设计领域还能够帮助定义API协议等。

建议 从零开始对JSON文档的内容进行声明非常麻烦也没有必要，建议先使用JSONSchema.net网站来根据已有的JSON文档生成对应的Schema文档，然后再根据具体的校验规则来逐步修缮。

基本示例

这里先给出一份简单的JSON文档和对该文档的JSON Schema描述，然后再介绍 JSON Schema的核心关键词 和具体规则。

JSON

```
# 文件名 demo.json 注意该行不作为json文件的内容

{
    "name": "wendingding",
    "age": 18,
    "email": "wendingding_ios@126.com",
    "height": 1.73,
    "isGoodMan": true,
    "tags": [
        "javaScript",
        "object-C",
        "C++",
        "swift",
        "php"
    ],
    "car": {
        "type": "A",
        "number": "粤A 66666",
        "price": 21344.88
    }
}
```

JSON

```
# 文件名 demo-schema.json 注意该行不作为json文件的内容
{
  "$schema": "http://json-schema.org/draft-07/schema#",

  "type": "object",
  "required": ["name","age","email","height","isGoodMan","tags","car"],
  "properties": {
    "name": {
      "type": "string",
      "examples": [ "wendingding" ],
    },
    "age": {
      "type": "integer",
      "examples": [ 18 ]
    },
    "email": {
      "type": "string",
      "examples": ["wendingding_ios@126.com"],
    },
    "height": {
      "type": "number",
      "examples": [ 1.73 ]
    },
    "isGoodMan": {
      "type": "boolean",
      "examples": [ true]
    },
    "tags": {
      "type": "array",
      "items": {
        "type": "string",
        "examples": ["javaScript", "object-C", "C++","swift","php"]
      }
    },
    "car": {
      "type": "object",
      "required": ["type","number", "price"],
      "properties": {
        "type": {
          "type": "string",
          "examples": ["A"]
        },
        "number": {
          "type": "string",
          "examples": ["粤A 66666"]
        },
        "price": {
          "type": "number",
          "examples": [21344.88 ]
        }
      }
    }
  }
}
```

推荐使用命令行工具的validate模块来使用Schema对准备好的JSON文档进行校验，列出执行细节：

```bash
wendingding$ npm install -g ujs-jsonvalidate

/usr/local/bin/validaten -> /usr/local/lib/node_modules/ujs-jsonvalidate/bin/validaten
/usr/local/bin/validate -> /usr/local/lib/node_modules/ujs-jsonvalidate/bin/validate
+ ujs-jsonvalidate@0.1.2
added 4 packages in 7.353s
wendingding$ validate demo.json demo-schema.json
JSON content in file demo.json is valid
```

因为在Schema文档中name字段是必要的，这里尝试删除demo.json文档中的 `name:"wendingding"` 部分，然后重新执行校验会发现提示错误信息。

```bash
wendingding$ cat demo.json
{
  "age": 18,
  "email": "wendingding_ios@126.com",
  "height": 1.73,
  "isGoodMan": true,
  "tags": [
    "javaScript",
    "object-C",
    "C++",
    "swift",
    "php"
  ],
  "car": {
    "type": "A",
    "number": "粤A 66666",
    "price": 21344.88
  }
}
wendingding$ validate demo.json demo-schema.json
Invalid: Missing required property: name
JSON Schema element: /required/0
JSON Content path:
```

核心关系词说明

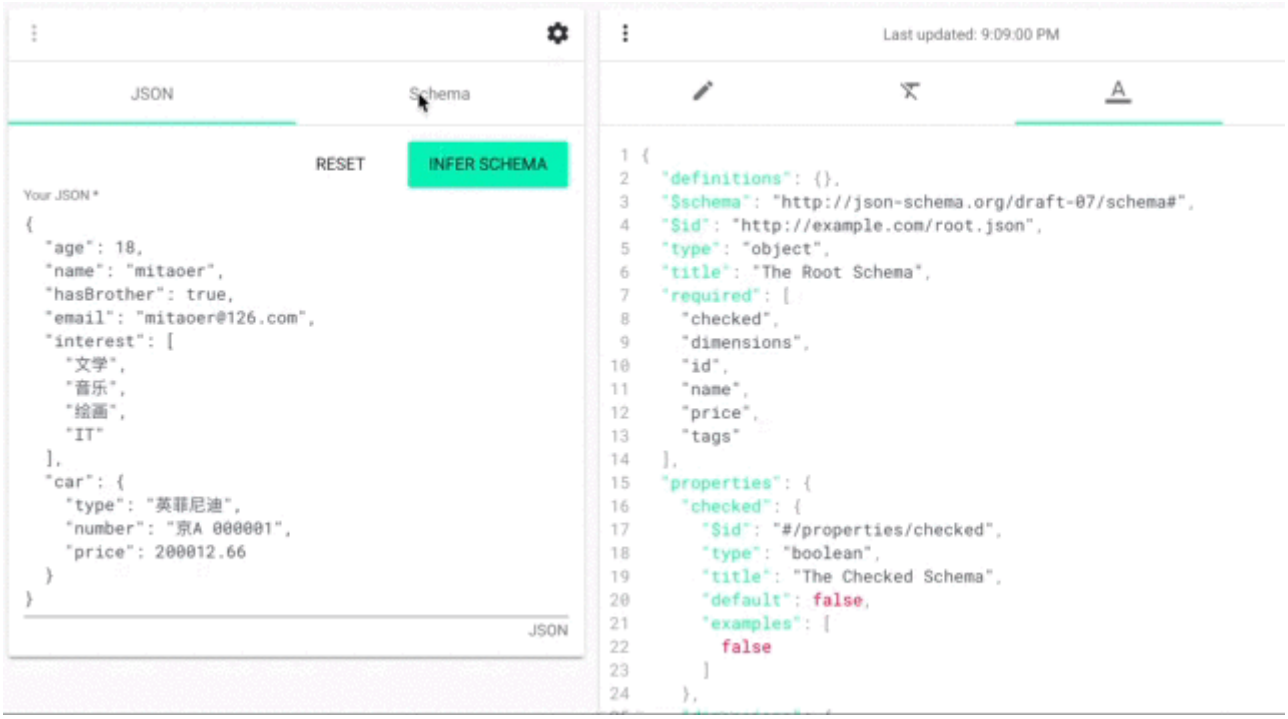
`type` 声明对应字段的类型。
`pattern` 使用正则表达式来限定字段的值。
`properties` 声明对象中的字段，其中包含具体字段的type值等信息。
`$schema` 声明遵循的JSON Schema标准版本，校验文档时使用该版本的规则。

`items` 如果字段是数组类型(array)，那么对数组元素的类型等进行限定。
`minimum | maximum` 如果是数值类型(number)，那么限定其取值的范围。
`minItems | maxItems` 用于校验数组成员的数目，设定最小数目和最大数目。
`examples` 提供该字段对应值的示例，在创建schema文档的时候通常根据JSON模板文件的内容生成。
`enum` 定义固定的枚举值来限制数组元素的取值，即数组的元素值只能是enum限定集合中的数据。

`additionalProperties` 将该字段设置为false可以禁止JSON文档当前节点中出现额外的字段。
`required` 该数组用于声明JSON文档中所有的必需字段，即必需包含这些字段，否则视为非法文档。
`dependencies` 设置字段的依赖关系，即JSON文档中出现了某个字段的出现必须依赖某个特定字段。
`patternProperties` 模式属性可以基于正则表达式来声明部分重复的字段名，如 `^string[1-3]$` 。

> **建议** 在工作中如果需要对JSON数据进行严格的语法和语义校验，那么建议先使用典型的JSON文档利用JSON Schema在线工具自动生成对应的Schema校验文件，然后再对该文件进行二次编辑，这样效率会更高一些。

这里给出 `JSON Schema在线工具` 的使用示例。



扩展 前文中在对JSON文档和对应Schema进行校验的时候，在命令行中使用的是 `validate模块` ，该模块是JSON Validate网站所对应的npm包，具体使用的是名为ujs-jsonvalidate的处理器。此外，ajv也是Node中优秀的一款JSON校验类库，它本身很简洁且兼容性很好，更多信息可以参考链接地址。