半小时实现一个 JSON 解析器

谢然
互联网行业 前端开发讲师
关注他

昨天在一个关于 JSON 语法设计的问题中讨论了一下为什么 JSON 格式要设计成那个样子。

我在回答中说，设计成那样是为了让它容易解析。

JSON 格式很容易解析，容易到什么程度呢？容易到一个极简的实现只需要半小时不到即可搞定，于是就有了这篇文章，半小时带你实现一个 JSON 解析器，真的只要半小时哦。

回顾一下我当时的回答，JSON 的设计让你可以在读到一个字符后就确定后面如何解析，具体来说：

当你在扫描文本流的过程中遇到一个左花括号（{）时，你知道你需要从当前字符开始，解析出一个对象，而解析对象的过程则是解析出多对 key/value，每解析完一对，如果你遇到了右花括号（}），则对这个对象的解析就结束了，而如果遇到的不是右花括号，则你需要解析下一对 key/value，直到遇见右花括号。而解析出一对 key/value，则是必须先解析出一个字符串，然后一个冒号（:），然后是这个 key 对应的**值**。

同理，当你遇到一个左方括号（[）时，你需要从这个字符开始解析出一个数组；而解析数组，则是解析出多个由逗号分隔的**值**，直到解析完一个值后遇到的不是逗号而是右方括号（]），则这个数组解析完毕。

而当你遇到一个双引号（"）时，则需要从当前位置开始解析出一个字符串。

遇到字母 “n” 的话，则是从当前位置开始往后读 4 个字符，且读到的 4 个字符组成的字符串必须是“null”，否则就应该报错。

遇到字母 “f” 的话，则是从当前位置开始往后读 4 个字符，且读到的 4 个字符组成的字符串必须是“null”，否则就应该报错。

遇到字母 “n” 的话，则是从当前位置开始往后读 5 个字符，且读到的 5 个字符组成的字符串必须是“false”，否则就应该报错。

剩下的就是数值了。

JSON 中对于以上几种类型都定义为值，而我们可能从任何位置开始遇到一个值，而在 JSON 中解析这个值只需要看其第一个字符是什么就可以了，这也是 JSON 容易解析的另一个原因，不像一般的编程语言，**JSON 的解析不需要先做 tokenize**。

为了减少复杂度，我们只解析没有任何多余空白内容的合法 JSON 字符串，暂时不考虑出错的问题，另外所有的 key 和字符串内都没有转义符，即形如以下：

```
var jsonStr = '{"a":1,"b":true,"c":false,"foo":null,"bar":[1,2,3]}'
var i = 0
```

知乎

然后我们先来实现一个解析出一个值的函数，名为 parseValue，同时，它解析完一个"值"后，会把 i 移动到下一个即将开始解析的位置，我们所有的函数都依赖这个全局变量i：

```
function parseValue() {
  if (str[i] === '{') {
    return parseObject()
  } else if (str[i] === '[') {
    return parseArray()
  } else if (str[i] === 'n') {
    return parseNull()
  } else if (str[i] === 't') {
    return parseTrue()
  } else if (str[i] === 'f') {
    return parseFalse()
  } else if (str[i] === '"') {
    return parseString()
  } else {//如果不考虑出错的话，不是以上所有的情况即
    return parseNumber()
  }
}
```

接下来，我们只需要一步步实现 parseValue 函数所调用到的这几个函数就行了，每个都非常容易，我就基本不写注释了：

```
// 所有的函数都是从i位置开始解析出一个对应类型的值
// 同时把i移动到解析完成后的下一个位置
function parseString() {
  var result = ''
  i++// 开始解析之前，i是指向字符开始的双引号的，但字符的内容是不包含这个双引号的
  while(str[i] != '"') {
    result += str[i++]
  }
  i++// 移动i到解析完成后的下一个位置
  return result
}

function parseNull() {
  // 简单粗暴，直接往后读出一个长度为4的个字符串出来
  // 如果不是null，则直接报错
  var content = str.substr(i, 4)

  if (content === 'null') {
    i += 4
    return null
  } else {
    throw new Error('Unexpected char at pos: ' + i)
  }
}

function parseFalse() {
  // 基本同上
  var content = str.substr(i, 5)

  if (content === 'false') {
    i += 5
    return false
  } else {
    throw new Error('Unexpected char at pos: ' + i)
  }
}

func
  //
```

知乎

```
  if (content === 'true') {
    i += 4
    return true
  } else {
    throw new Error('Unexpected char at pos: ' + i)
  }
}

function parseNumber() {
  // 本函数的实现并没有考虑内容格式的问题，实际上JSON中的数值需要满足一个格式
  // 不过好在这个格式基本可以用正则表达出来，不过这里就不写了
  // 想写的话对着官网的铁路图写一个出来就行了
  // 并且由于最后调用了parseFloat，所以如果格式不对，还是会报错的
  var numStr = ''//-2e+8
  // 此处只要判断i位置还是数字字符，就继续读
  // 为了方便，写了另一个helper函数
  while (isNumberChar(str[i])) {
    numStr += str[i++]
  }
  return parseFloat(numStr)
}

// 判断字符c是否为组成JSON中数值的符号
function isNumberChar(c) {
  var chars = {
    '-': true,
    '+': true,
    'e': true,
    'E': true,
    '.': true
  }
  if (chars[c]) {
    return true
  }
  if (c >= '0' && c <= '9') {
    return true
  }
  return false
}

// 解析数组，就很容易了
// 掐头去尾
// 然后一个值一个逗号
// 如果解析完一个值后没遇到逗号，说明解析完了
// 现在你知道没有多余的逗号有多好解析了吧~
function parseArray() {
  i++
  var result = []//[1234,"lsdf",true,false]
  while(str[i] !== ']') {
    result.push(parseValue())
    if (str[i] === ',') {
      i++
    }
  }
  i++
  return result
}

// 解析对象，一如既往的简单
// 掐头去尾
// 然后一个key，是字符串
// 一个冒号
// 一个值，可能是任意类型，所以调用parseValue
// 最后，如果解析完一组k/v对，遇到了逗号，则解析下一组，没遇到逗号，则解析完毕
func
  i+
```

知乎

```
    var key = parseString()
    i++//由于只考虑合法且无多余空白的JSON，所以这里就不判断是不是逗号了，正常应该是发现不是
    var value = parseValue()
    result[key] = value
    if (str[i] === ',') {
      i++
    }
  }
  i++
  return result
}
```

最后，**JSON 数据的整个内容其实就表示了一个值**，所以我们只要把 i 置为 0，然后从头开始解析出来一个**值**就完事了！

```
function parse(json) {
  i = 0
  jsonStr = json
  return parseValue()
}
```

怎么样，是不是只花半小时就写出来了。

现在动手自己写一个吧~不要看我的代码。

最后，我们来简单分析一下，上面的代码虽然没有**直接的出现递归调用**，但实际上发生了**间接的递归**：parseValue调用了**parseObject**与parseArray，而这两个函数在内又调用了parseValue。

解析过程是递归的，所以我们可以得出结论，即 **JSON 数据是递归的**。

并且，JSON 格式里是没有环的，**实际上，JSON 格式表达了一颗树，一颗多叉树！**

然后，**留个作业**，写完 parse，试着写一个 JSON.stringify 出来！

或者，试着让自己写出来的 parse 功能能够**处理 JSON 字符串中多余的空白**。

再然后，让 parseString **支持解析出转义字符及 Unicode 转义符**：\u6211。

再然后，额，好像没有然后了，再然后，就是把你写的这个整一整，再加些测试用例，放到github上面，命名为 JSON5.js，然后你就可以在简历上写上实现过 JSON 解析器了！

再然后，你可以试着实现一个解析**函数调用表达式**的解析器：

sum(sqrt(pow(3,4)),div(7,8),add(9,1))

我看好你！！

发布于 2017-07-24 00:29

前端开发 JavaScript JSON

发布一条带图评论吧

35

知乎

.....反斜杠是个大问题你居然就直接 暂不考虑 了.....

2017-07-24 回复 4

谢然 作者

并不算大问题，处理反斜杠的逻辑跟这里其它的函数差不多，就多写点代码而已，因为反斜杠后面要么一个字符，要么\u后面跟4个字符

2017-07-24 回复 2

Lyp

的确如此，反斜杠是最容易的，因为必定包含在string里，对string遍历就行

2020-06-16 回复 喜欢

德鲁大叔

纸上得来终觉浅，觉知此事要躬行。

2017-07-25 回复 4

好玩意不好用

在C/C++世界里面，说到好用的 json 解析器那就非 zua 莫属了，支持任意嵌套、null、false、true完整支持，编码舒适，非常简单易用。

Zua: github.com/liqiongfan/Z...

有好的改进意见或者用法可以及时反馈，或许下一个特性就是你提出的哦。

2021-08-04 回复 1

南山

能给源码链接吗，抄了代码运行不起来

2017-07-25 回复 2

谢然 作者

调试一下看看报什么错啊

2017-07-25 回复 1

3DXcat

一年了，还没改

2019-07-20 回复 2

好玩意不好用

在C/C++世界里面，说到好用的 json 解析器那就非 zua 莫属了，支持任意嵌套、null、false、true完整支持，编码舒适，非常简单易用。

Zua: github.com/liqiongfan/Z...

有好的改进意见或者用法可以及时反馈，或许下一个特性就是你提出的哦。

2021-08-04 回复 喜欢

裂章

解析true那里写成null了。

2017-07-24 回复 2

风中的雪糕 ▸ 谢然

3天了,你还没改[鄙视]

2017-07-27 回复 4

送命猫

4年了

2021-01-13 回复 3

查看全部 10 条回复 》

sunsoul

解析true和false那儿跟null串了...

2017-07-24 回复 1

知乎

是说注释吗，我的意思就是逻辑基本同null的解析

2017-07-25 回复 1

展开其他 1 条回复 ›

alexpeng

goo

2017-07-24 回复 1

jello chen

如果考虑完整实现的话，还是有很多要做的

2017-07-24 回复 1

谢然 作者

那是必须的，各种错误都得处理。不过我基本上也都在注释里列出来了

2017-07-24 回复 1

非我梦想

答主说的解析json以及末尾解析数字，我都自己实现过（没有参考答主代码哦），那我是不是也可以在简历上写我实现过json解析器，hh

2023-08-01 回复 喜欢

知乎用户uXVF4k

别人写了一晚，我写了两天。你用半个小时，太优秀了。

2021-09-26 回复 喜欢

渺小的人

听他吹牛逼

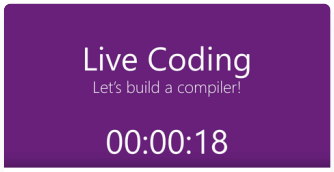
2022-08-08 回复 喜欢

点击查看全部评论 ›

发布一条带图评论吧

JsonPath —— JSON 解析神器

极乐君 发表于极乐科技



.NET Core3发布Json API（译文）

森林蝙蝠

jsmn | 一个资源占用极小，解析速度最快的json解析器

嵌入式开源项目精选专栏本专栏由Mculover666创建，主要内容为寻找嵌入式领域内的优质开源项目，一是帮助开发者使用开源项目实现更多的功能，二是通过这些开源项目，学习大佬的代码及背后的...

mculo... 发表于嵌入式开源...

【酷Go推荐】让 json 解析更简单高效的 GJSON

Golang | Gjson 库简介 什么是 Gjson： GJSON 是一个 Golang 包，它提供了一种快速，简单的方法来从 json 格式文档中获取值。拥有比如单行检索，用 "."符号来寻找路径，送

asta谢 发表于GoCN社