# Json

由 enlai.feng创建于三月 10, 2024

## 什么是Json？JavaScript Object Notation

JSON（JavaScript Object Notation）文件是一种轻量级的数据交换格式，其作用可以从几个不同的角度来理解：

| 作用 | 描述 |
| --- | --- |
| 数据交换 | JSON是一种文本格式，它使得数据可以轻松地在不同的系统之间进行交换。它是独立于语言的，但使用了类似于JavaScript的约定，这使得许多编程环境能够轻松地解析和生成JSON数据。 |
| **配置文件** | JSON文件常被用作配置文件。例如，许多Web应用程序和开发工具使用`package.json`文件来存储项目的配置信息，如依赖项、脚本和其他设置。 |
| **数据存储** | 虽然JSON不是为数据库设计的，但它也常用于存储数据。某些NoSQL数据库，如MongoDB，内部就是使用类似JSON的格式（如BSON）来存储数据。 |
| **序列化和反序列化** | 在编程中，JSON用于序列化（将数据结构或对象状态转换为可存储或传输的格式）和反序列化（从序列化格式恢复到数据结构）。 |
| | |
| | |

## Json语法

JSON语法是javaScript语法的子集，**javaScript用[]中括号来表示数组，用{}大括号来表示对象，JSON亦是如此。**

### Json数组：

```
var employees = [
    { "firstName":"Bill" , "lastName":"Gates" },
    { "firstName":"George" , "lastName":"Bush" },
    { "firstName":"Thomas" , "lastName": "Carter" }
    ];
```

### Json对象

```
var obj = {

            age: 20,
            str: "zhongfucheng",
            method: function () {
                alert("我爱学习");
            }

        };
```

**数组可以包含对象，在对象中也可以包含数组。**

## Json语法解析

javaScript原生支持JSON的，**我们可以使用eval()函数来解析JSON，把JSON文本数据转换成一个JavaScript对象。**

```
function test() {
            //在写JOSN的时候，记得把带上逗号
```

```
            var txt = "{a:123," +
                    "b:'zhongfucheng'}";

            //使用eval解析JSON字符串，需要增添()
            var aa = eval("(" + txt + ")");
            alert(aa);

        }
```

### 读取Json文件内容

假设我们有一个名为`example.json`的文件，其内容如下：

```
{
    "name": "John Doe",
    "age": 30,
    "is_student": false,
    "scores": [95, 85, 90],
    "address": {
        "street": "123 Main St",
        "city": "Anytown",
        "zip": "12345"
    }
}
```

这个JSON文件描述了一个人的基本信息，包括姓名、年龄、是否是学生、一组分数以及一个地址对象。

现在，我们将使用`cJSON`库来读取和解析这个JSON文件。以下是C语言代码示例：

```
#include <stdio.h>
#include <stdlib.h>
#include "CJSON.h"

int main()
{
    //打开JSON文件
    FILE *file =
}
```

C语言文件

```
#include <stdio.h>
#include <stdlib.h>
#include "cJSON.h"

int main()
{
    //打开JSON文件
    FILE *file = fopen("example.json", "r");
    if(file == NULL)
    {
        perror("Error opening file");
        return -1;
    }

    // 读取文件内容到字符串
    fseek(file, 0, SEEK_END);   //将文件指针移动到文件末尾
    long length = ftell(file);
    fseek(file, 0, SEEK_SET);
    char *data = (char*)malloc(length + 1);
```

```
    if (data == NULL) {
        fprintf(stderr, "Malloc failed\n");
        fclose(file);
        return -1;
    }

    fread(data, 1, length, file);
    data[length] = '\0'; // 确保字符串以NULL结尾

    // 关闭文件
    fclose(file);

    // 解析JSON数据
    cJSON *json = cJSON_Parse(data);
    if (json == NULL) {
        const char *error_ptr = cJSON_GetErrorPtr();
        if (error_ptr != NULL) {
            fprintf(stderr, "Error before: %s\n", error_ptr);
        }
        free(data);
        return -1;
    }

    // 读取"name"字段
    cJSON *name = cJSON_GetObjectItemCaseSensitive(json, "name");
    if (cJSON_IsString(name) && (name->valuestring != NULL)) {
        printf("Name: %s\n", name->valuestring);
    }

    // 读取"age"字段
    cJSON *age = cJSON_GetObjectItemCaseSensitive(json, "age");
    if (cJSON_IsNumber(age)) {
        printf("Age: %d\n", age->valueint);
    }

    // 读取"address"对象中的"city"字段
    cJSON *address = cJSON_GetObjectItemCaseSensitive(json, "address");
    cJSON *city = cJSON_GetObjectItemCaseSensitive(address, "city");
    if (cJSON_IsString(city) && (city->valuestring != NULL)) {
        printf("City: %s\n", city->valuestring);
    }

    // 清理JSON对象
    cJSON_Delete(json);
    // 释放读取的数据
    free(data);

    return 0;
}
```

无标签