JSON

JavaScript Object Notation

Introduction to JSON

- JSON is a popular format for data interchange over the Internet. It is supported by many programming languages:
  - ActionScript, C, C#, Clojure, ColdFusion, Common Lisp, E, Erlang, Go, Java, JavaScript, Lua, Objective CAML, Perl, PHP, Python, Rebol, Ruby, Scala, and Scheme.
- JSON can represent four primitive types (strings, numbers, booleans, and null) and two structured types (objects and arrays).
- An object is an unordered collection of zero or more name/value pairs, where a name is a string and a value is a string, number, boolean, null, object, or array..
- [YouTube] Learn JSON in 10 Minutes
  - https://www.youtube.com/watch?v=iiADhChRriM

JSON Data Types

- String
  - "SAN FRANCISCO"
- Number
  - 100, 3.14, 3e10
- Boolean
  - true, false
- null
- Array
  - [1, 3, 5, 7]
- Object
  - { "Width": 800, "Height": 600 }

10. Create a JSON object from a string

```
#include <iostream>
#include <json.hpp>
using nlohmann::json;
using std::cout;
using std::cin;
using std::endl;

int main()
{
   JSON ex1 = JSON::parse("[1,2,3]");
   cout << ex1[1] << endl;
   return 0;
}
```

1. More complex structure

```
#include <iostream>
#include <json.hpp>
using nlohmann::json;
using std::cout;
using std::cin;
using std::endl;

int main()
{
    json ex1 = json::parse( "{ \"pi\": 3.14159 }" );
    json ex2 = json::parse(R"(
    {
      "pi": 3.14159,
      "happy": true
    }
    )");
    cout << ex1["pi"] << endl;
    cout << ex2["pi"] << endl;
    return 0;
}
```

Compile with the option

```
g++ j1.cpp -I ~solomon/CPP
```

You save the trouble of escaping every quotation mark in the raw string.

Raw String

```
#include <iostream>
#include <string>
using std::cout;
using std::endl;

int main() {
    std::string s1 = R"({ "A": "Alice", "B": "Bob" })";
    cout << "s1 = " << s1 << endl;
    return 0;
}
```

8. Raw String

```
#include <iostream>
#include <json.hpp>
#include <string>      // Raw string is defined here.
using std::cout;
using std::endl;
using nlohmann::json;

int main() {
   std::string s1 = R"({ "A": "Alice", "B": "Bob" })";
   std::string s2 = "{ \"A\": \"Alice\", \"B\": \"Bob\" }";
   cout << "s1 = " << s1 << endl
      << "s2 = " << s2 << endl;
   json j1 = json::parse( R"( { "A": "Alice", "B": "Bob" } )" );
   json j2 = json::parse( R"( { "B": "Bob", "A": "Alice" } )" );
   cout << j1.dump() << endl;
   cout << j2.dump() << endl;
   if (j1 == j2) cout << "These two JSON objects are equal.\n";
   return 0;
}
```

2. Dump a JSON object to a string

```
#include <iostream>
#include <string>
#include <json.hpp>
using nlohmann::json;
using std::cout;
using std::endl;
using std::string;

int main()
{
   json ex2;
   ex2["A"] = { 1, 2, 3 };
   string s = ex2.dump();
   cout << s << endl;
   return 0;
}
```

3. Write a JSON object to a file

```
#include <iostream>
#include <fstream>
#include <json.hpp>
using nlohmann::json;
using std::cout;
using std::endl;

int main()
{
    json ex1 = json::parse(R"(
    {
      "pi": 3.14159,
      "happy": true
    }
    )");

    std::ofstream f("j3.json");
    f << std::setw(4) << ex1 << std::endl;
    f.close();
    return 0;
}
```

4. Read a JSON object from a file

```
#include <iostream>
#include <fstream>
#include "json.hpp"
using nlohmann::json;

int main()
{
    json ex4;
    std::ifstream f("j3.json");
    f >> ex4;
    f.close();
    std::cout << ex4["pi"] << std::endl;
    return 0;
}
```

5. Enumerate all entries

```
#include <iostream>
#include <json.hpp>
using nlohmann::json;
using std::cout;
using std::endl;

int main()
{
    json j = json::parse( R"( {"A": "Alfa", "B": "Bravo", "C": "Charlie"} )" );
    for (json::iterator it = j.begin(); it != j.end(); ++it)
        cout << it.key() << " : " << it.value() << endl;
    for (auto& element : j.items())
        cout << element.key() << " : " << element.value() << endl;
    for (auto& [key, value] : j.items())
        cout << key << " : " << value << endl;
    return 0;
}
```

6. find(), count()

```
#include <iostream>
#include <json.hpp>
using nlohmann::json;
using std::cout;
using std::endl;

int main()
{
   json j = json::parse( R"( {"A": "Alfa", "B": "Bravo", "C": "Charlie"} )" );
   if (j.contains("A")) cout << "j contains A" << endl;
   if (j.find("B") != j.end()) cout << "j contains B" << endl;
   if (j.count("C") > 0) cout << "j contains C" << endl;
   j.erase("B");      // delete an entry
   cout << j.dump() << endl;
   return 0;
}
```

7. Automatic conversion from vector and map

```
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <json.hpp>
using std::vector;
using std::map;
using std::string;
using nlohmann::json;
using std::cout;
using std::endl;

int main()
{
    vector<int> A {1, 2, 3, 4};
    map<string, int> B { {"Alice", 1}, {"Bob", 2}, {"Carol", 3} };
    json j1(A);
    json j2 = B;
    cout << j1.dump() << endl;
    cout << j2.dump() << endl;
    return 0;
}
```

9. diff(), patch()

```
#include <iostream>
#include <json.hpp>
#include <string>
using std::cout;
using std::endl;
using nlohmann::json;

int main() {
    json original = json::parse( R"( { "A": "Alice", "B": "Bob" } )" );
    json goal = json::parse( R"( { "B": "Bob", "C": "Carol" } )" );
    json difference = json::diff(original, goal);
    cout << difference.dump() << endl;
    json j3 = original.patch(difference);
    cout << j3.dump() << endl;
    json p = json::parse( R"([
      { "op": "replace", "path": "/C", "value": "Charlie" }
    ])" );
    json j4 = goal.patch(p);
    cout << j4.dump() << endl;
    return 0;
}
```

References

- [GitHub] nlohmann json
  - https://github.com/nlohmann/json
- RFC 8259 - The JavaScript Object Notation (JSON) Data Interchange Format, December 2017.
  - This documents obsoletes RFC 7159, which obsoletes RFC 4627 (July 2006).
- RFC 6901 - JSON Pointer
- RFC 6902 - JSON Patch
- RFC 7386 - JSON Merge Patch