JSON Basics

JSON stands for JavaScript Object Notation. JSON is an open standard for data interchange that is formatted in a way easy for humans to read and understand. JSON objects are entirely text based, and are language independent, though it uses conventions that will be familiar to programmers with familiarity in the C-family of languages, like us.

JSON is built on two main structures. The first is an object. Each object consists of some name/value pairs, separated by commas, and enclosed by curly braces. An object could take on the following forms, among others:

```
{“First Name”: “Dr”,
“Last Name”: “Chamberland”}

{“Grade”: 100}                          // name/value pair: ”name” -> “Grade”, “value” -> 100

{“Winning”: true}
```

The name element in the pair must be a string. The value element can take on a number of different types. The value can be a number, such as an int or double precision number. If the value is a string it must be enclosed in double quotes. The value can also be a Boolean value (true or false), the null value, another object itself, or an array of objects, which brings us to the second fundamental structure of JSON, the array.

An array is an ordered collection of values. Like the name/value pairs in an object, values in an array are separated by a comma. Again, the values can take on any of the following types: string, number, object, array, Boolean true or false, and null. These structures can be nested within one another as well. Some examples of JSON arrays could look like this:

```
[100,200,300,400,500]                   //Array of numbers

[
 { “First Name” : “John”,
   “Last Name”: “Doe”  },
 {  “First Name”: “Jane”,
   “Last Name” : “Smith”}
]                                       //Array of objects
```

Many times JSON data takes on a hierarchical structure as many objects, arrays, and values are nested within one another. Note again that the “value” in a name/value pair can be another complete object or even array of objects. This is useful for items which have many sub-groups underneath the same heading as well as for keeping data organized. For example, one might want to sort data points from different devices, or with different time stamps, etc… and the nesting of objects and arrays allows easy manipulation of data while maintaining organization.

Nested Object example

```
{
  "geo location" : {
              "latitude" : {
                     "degrees" : 45,
                     "minutes" : 56,
                     "seconds" : 5
              },
              "longitude" : {
                     "degrees" : 34,
                     "minutes" : 12,
                     "seconds" : 2
              }
}
```

Since JSON is purely a text format, JSON objects can even be written and stored in .txt files, meaning it is possible for any system to read and write JSON objects without additional software. This makes data sharing across platforms easier, as one can just write their data to a txt file if they want or need to and the JSON parser on the other end will be able to read the data.

Applications

One of the most common uses of JSON is to fetch JSON data from a server and convert the data to a JavaScript object for use in a webpage. Most web browsers nowadays have built in JSON parsing capabilities. These browsers include internet explorer, google chrome, firefox, opera, and safari. JSON is similar to XML but it is smaller, faster, and easier to parse.

Many websites use RSS (Rich Site Summary) feeds with JSON to share data because JSON feeds can be loaded asynchronously more easily than RSS with XML. Asynchronous I/O allows other processing to continue before its transmission has finished, meaning the data exchange can happen in the background without blocking other function so webpages can do things like change content and graphics on the page without requiring a page refresh from the user. JSON is also directly supported in JavaScript applications with no additional software needed.

There are also JSON parsers/writers available for C++. These libraries have various built in functions that allow one to manipulate the data in the JSON file. For instance, there might be a function to get the value from the name/value pairs in an object, write objects to an array, or add other functionality. There are many JSON libraries available. We have some example code using rapidjson, but there are many more including jsoncpp, JSON++, and others.

Additional Resources

json.org
http://www.thomaswhitton.com/blog/2013/06/27/json-c-plus-plus-examples/
http://www.copterlabs.com/blog/json-what-it-is-how-it-works-how-to-use-it/
http://www.w3schools.com/json/
http://en.wikipedia.org/wiki/JSON