An Introduction to JSON 15th Day 15%

JSON, short for **JavaScript Object Notation** is a format for sharing data. As its name suggests, JSON is derived from the JavaScript programming language, but it's available for use by many languages including Python, Ruby, PHP, and Java. JSON is usually pronounced like the name "Jason."

JSON uses the .json extension when it stands alone. When it's defined in another file format (as in .html), it can appear inside of quotes as a JSON string, or it can be an object assigned to a variable. This format is easy to transmit between web server and client or browser.

Very readable and lightweight, JSON offers a good alternative to XML and requires much less formatting. This informational guide will get you up to speed with the data you can use in JSON files, and the general structure and syntax of this format.

Syntax and Structure.

A JSON object is a key–value data format that is typically rendered in curly braces. When you're working with JSON, you'll likely see JSON objects in a .json file, but they can also exist as a JSON object or string within the context of a program.

A JSON object looks something like this:

```
{
  "first_name" : "Samaiya",
  "last_name" : "Joan",
  "location" : "Nigeria",
  "online" : true,
  "followers" : 1458
}
```

Although this is a very short example, and JSON could be many lines long, this shows that the format is generally set up with two curly braces (or curly brackets) that look like this {} on either end of it, and with key–value pairs populating the space between. Most data used in JSON ends up being encapsulated in a JSON object.

Key–value pairs have a colon between them as in “key” : “value”. Each key–value pair is separated by a comma, so the middle of a JSON looks like this: "key" : "value", "key" : "value", "key": "value". In our example above, the first key–value pair is "first_name" : "Samaiya".

JSON **keys** are on the left side of the colon. They need to be wrapped in double quotation marks, as in “key”, and can be any valid string. Within each object, keys need to be unique. These key strings can include white spaces, as in "first name", but that can make it harder to access when you’re programming, so it’s best to use underscores, as in "first_name".

JSON values are found to the right of the colon. At the granular level, these need to be one of 6 simple data types:

•strings
•numbers
•objects
•arrays
•Booleans (true or false)
•null

Complex Types in JSON.

JSON can store nested objects in JSON format in addition to nested arrays. These objects and arrays will be passed as values assigned to keys, and typically will be comprised of key–value pairs as well.

Nested Objects.

In the users.json file below, for each of the four users ("sammy", "jesse", "drew", "jamie") there is a nested JSON object passed as the value for each of the users, with its own nested keys of "username" and "location" that relate to each of the users. The first nested JSON object is highlighted below.

```
{
  "sammy" : {
    "username"  : "SammyShark",
    "location"  : "Indian Ocean",
    "online"    : true,
    "followers" : 987
  },
  "jesse" : {
    "username"  : "JesseOctopus",
    "location"  : "Pacific Ocean",
    "online"    : false,
    "followers" : 432
  },
  "drew" : {
    "username"  : "DrewSquid",
    "location"  : "Atlantic Ocean",
    "online"    : false,
    "followers" : 321
  },
  "jamie" : {
    "username"  : "JamieMantisShrimp",
    "location"  : "Pacific Ocean",
    "online"    : true,
    "followers" : 654
  }
}
```

In the use.json file above, curly braces are used throughout to form a nested JSON object with associated username and location data for each of four users. Just like any other value, when using objects, commas are used to separate elements.

Nested Arrays.

Data can also be nested within the JSON format by using JavaScript arrays that are passed as a value. JavaScript uses square brackets [ ] on either end of its array type. Arrays are ordered collections and can contain values of differing data types.

We may use an array when we are dealing with a lot of data that can be easily grouped together, like when there are various websites and social media profiles associated with a single user.

With the first nested array highlighted, a user profile for Sammy may look like this:

```
{
  "first_name" : "Sammy",
  "last_name" : "Shark",
  "location" : "Ocean",
  "websites" : [
    {
      "description" : "work",
      "URL" : "https://www.digitalocean.com/"
    },
    {
      "desciption" : "tutorials",
      "URL" : "https://www.digitalocean.com/community/tutorials"
    }
  ],
  "social_media" : [
    {
      "description" : "twitter",
      "link" : "https://twitter.com/digitalocean"
    },
    {
      "description" : "facebook",
      "link" : "https://www.facebook.com/DigitalOceanCloudHosting"
    }
  ]
}
```

The "websites" key and “social_media" key each use an array to nest information belonging to Sammy’s 2 website links and 3 social media profile links. We know that those are arrays because of the use of square brackets.

Using nesting within our JSON format lets us work with more complicated and hierarchical data.

Comparison to XML

XML, or eXtensible Markup Language, is a way to store accessible data that can be read by both humans and machines. The XML format is available for use across many programming languages.

In many ways, XML is very similar to JSON, but it requires much more text so is longer in length and more time consuming to read and write. XML must be parsed with an XML parser, but JSON can be parsed with a standard function. Also unlike JSON, XML cannot use arrays.

We’ll look at an example of the XML format and then look at the same data rendered in JSON.

```
<users>
    <user>
        <username>SammyShark</username> <location>Indian Ocean</location>
    </user>
    <user>
        <username>JesseOctopus</username> <location>Pacific Ocean</location>
    </user>
    <user>
        <username>DrewSquir</username> <location>Atlantic Ocean</location>
    </user>
    <user>
        <username>JamieMantisShrimp</username> <location>Pacific Ocean</location>
    </user>
</users>
```

```
{"users": [
  {"username" : "SammyShark", "location" : "Indian Ocean"},
  {"username" : "JesseOctopus", "location" : "Pacific Ocean"},
  {"username" : "DrewSquid", "location" : "Atlantic Ocean"},
  {"username" : "JamieMantisShrimp", "location" : "Pacific Ocean"}
] }
```

We see that JSON is much more compact and does not require end tags while XML does. Additionally, XML is not making use of an array as this example of JSON does (which you can tell through the use of square brackets).

If you are familiar with HTML, you'll notice that XML looks quite similar in its use of tags. While JSON is leaner and less verbose than XML, and quick to use in many situations including AJAX applications, you'll want to understand the type of project you're working on before deciding what data structures to use.

Read more about JSON @ http://www.json.org/

Best Wishes.

Lux Tech Academy.