How-To: Work with JSON in JavaScript

Introduction

Because **JSON** is derived from the JavaScript programming language, it is a natural choice to use as a data format in JavaScript. JSON, short for **JavaScript Object Notation**, is usually pronounced like the name "Jason."

To learn more about JSON in general terms, read the "An Introduction to JSON" tutorial.

To begin thinking about where you may use JSON in your JavaScript programs, some general use cases of JSON include:

- Storing data
- Generating data structures from user input
- Transferring data from server to client, client to server, and server to server
- Configuring and verifying data

This tutorial will provide you with an introduction to working with JSON in JavaScript. To make the most use of this introduction, you should have some familiarity with the JavaScript programming language.

JSON Format

JSON's format is derived from JavaScript object syntax, but it is entirely text-based. It 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. Read more about the syntax and structure here.

When you're working with a `.json` file, it will look like this:

sammy.json

```
{
  "first_name"  :  "Sammy",
  "last_name"   :  "Shark",
  "online"      :  true
}
```

Copy

If, instead, you have a JSON object in a `.js` or `.html` file, you'll likely see it set to a variable:

```
var sammy = {
  "first_name"  :  "Sammy",
  "last_name"   :  "Shark",
  "online"      :  true
}
```

Copy

Additionally, you may see JSON as a string rather than an object within the context of a JavaScript program file or script. In this case, you may also see it all on one line:

```
var sammy = '{"first_name" : "Sammy", "last_name" : "Shark", "location" : "Ocean"}';
```

Copy

Converting JSON objects into strings can be particularly useful for transporting data in a quick manner.

We've gone over the general format of JSON and how you may expect to see it as a `.json` file, or within JavaScript as an object or a string.

Comparison to JavaScript Object

It is worth keeping in mind that JSON was developed to be used by any programming language, while JavaScript objects can only be worked with directly through the JavaScript programming language.

In terms of syntax, JavaScript objects are similar to JSON, but the keys in JavaScript objects are not strings in quotes. Also, JavaScript objects are less limited in terms of types passed to values, so they can use functions as values.

Let's look at an example of a JavaScript object of the website user Sammy Shark who is currently online.

```
var user = {
    first_name: "Sammy",
    last_name : "Shark",
    online    : true,
    full_name : function() {
       return this.first_name + " " + this.last_name;
    }
};
```

Copy

This will look very familiar to you as a JSON object, but there are no quotes around any of the keys (`first_name`, `last_name`, `online`, or `full_name`), **and** there is a function value in the last line.

If we want to access the data in the JavaScript object above, we could use dot notation to call `user.first_name;` and get a string, but if we want to access the full name, we would need to do so by calling `user.full_name();` because it is a function.

JavaScript objects can only exist within the JavaScript language, so when you're working with data that needs to be accessed by various languages, it is best to opt for JSON.

Accessing JSON Data

JSON data is normally accessed in Javascript through dot notation. To understand how this works, let's consider the JSON object `sammy`:

```
var sammy = {
  "first_name"  :  "Sammy",
  "last_name"   :  "Shark",
  "online"      :  true
}
```

Copy

In order to access any of the values, we'll be using dot notation that looks like this:

```
sammy.first_name
sammy.last_name
sammy.online
```

Copy

The variable `sammy` is first, followed by a dot, followed by the key to be accessed.

To create a JavaScript alert that shows us the value associated with the key `first_name` in a pop-up, we can do so by calling the JavaScript `alert()` function:

```
alert(sammy.first_name);
```

Copy

```
Output
Sammy
```

Here, we've successfully called the value associated with the `first_name` key from the `sammy` JSON object.

We can also use square bracket syntax to access data from JSON. To do that, we would keep the key in double quotes within square brackets. For our `sammy` variable above, using square bracket syntax in an `alert()` function looks like this:

```
alert(sammy["online"]);
```

Copy

```
Output
true
```

When you're working with nested array elements, you should call the number of the item in your array. Let's consider the JSON below:

```
var user_profile = {
  "username" : "SammyShark",
  "social_media" : [
    {
      "description" : "twitter",
      "link" : "https://twitter.com/digitalocean"
    },
    {
      "description" : "facebook",
      "link" : "https://www.facebook.com/DigitalOceanCloudHosting"
    },
    {
      "description" : "github",
      "link" : "https://github.com/digitalocean"
    }
  ]
}
```

Copy

To access the string `facebook`, we can call that item in the array within the context of dot notation:

```
alert(user_profile.social_media[1].description);
```

Copy

```
Output
facebook
```

Notice that for each nested element we'll use an additional dot.

Using dot notation or square bracket syntax allows us to access the data contained in JSON format.

Functions for Working with JSON

This section will look at two methods for stringifying and parsing JSON. Being able to convert JSON from object to string and vice versa is useful for transferring and storing data.

JSON.stringify()

The `JSON.stringify()` function converts an object to a JSON string.

Strings are useful for transporting data from a client to a server through storing or passing information in a lightweight way. For example, you may gather a user's settings on the client side and then send them to a server. Later, you can then read the information with the `JSON.parse()` method and work with the data as needed.

We'll look at a JSON object that we assign to the variable `obj`, and then we'll convert it using `JSON.stringify()` by passing `obj` to the function. We can assign this string to the variable `s`:

```
var obj = {"first_name" : "Sammy", "last_name" : "Shark", "location" : "Ocean"}
var s = JSON.stringify(obj)
```

Copy

Now, if we work with `s`, we'll have the JSON available to us as a string rather than an object.

```
'{"first_name" : "Sammy", "last_name" : "Shark", "location" : "Ocean"}'
```

Copy

The `JSON.stringify()` function lets us convert objects to strings. To do the opposite, we'll look at the `JSON.parse()` function.

JSON.parse()

Strings are useful for transporting but you'll want to be able to convert them back to a JSON object on the client and/or the server side. We can do this using the `JSON.parse()` function.

To convert the example in the JSON.stringify() section above, we would pass the string `s` to the function, and assign it to a new variable:

```
var o = JSON.parse(s)
```

Copy

Then, we would have the object `o` to work with, which would be identical to the object `obj`.

To take a deeper look, let's consider an example of `JSON.parse()` within the context of an HTML file:

```
<!DOCTYPE html>
<html>
<body>

<p id="user"></p>

<script>
var s = '{"first_name" : "Sammy", "last_name" : "Shark", "location" : "Ocean"}';
var obj = JSON.parse(s);

document.getElementById("user").innerHTML =
"Name: " + obj.first_name + " " + obj.last_name + "<br>" +
"Location: " + obj.location;
</script>

</body>
</html>
```

Copy

```
Output
Name: Sammy Shark
Location: Ocean
```

Within the context of an HTML file, we can see how the JSON string `s` is converted to an object that is retrievable on the final rendering of the page by accessing the JSON via dot notation.

`JSON.parse()` is a secure function to parse JSON strings and convert them to objects.

Conclusion

JSON is a natural format to use in JavaScript and has many implementations available for use in many popular programming languages. If you want to use the format in another progamming language, you can see full language support on the "Introducing JSON" site.

Because it is lightweight and is readily transferred between programming languages and systems, JSON has been experiencing increased support in APIs, including the Twitter API.

You likely won't be creating your own `.json` files but procuring them from other sources. You can check out these resources to learn about converting other data structures to JSON.

Courtesy: https://www.digitalocean.com/community/tutorials/how-to-work-with-json-in-javascript

Modified: 2021.10.04.1.50.PM

Dököll Solutions, Inc.