Parse JSON Object with Java Script

JSON, or JavaScript Object Notation, is all around us. If you've ever used a web app, there's a very good chance that it used JSON to structure, store, and transmit data between its servers and your device.

In this article, we'll briefly go over the differences between JSON and JavaScript, then jump into different ways to parse JSON with JavaScript in the browser and in Node.js projects.

Differences between JSON and JavaScript

While JSON looks like regular JavaScript, it's better to think of JSON as a data format, similar to a text file. It just so happens that JSON is inspired by JavaScript syntax, which is why they look so similar.

Let's take a look at JSON objects and JSON arrays and compare them to their JavaScript counterparts.

JSON objects vs JavaScript Object Literals

First, here's a JSON object:

```
{
   "name":  "Jane Doe",
   "favorite-game":  "Stardew Valley",
   "subscriber":  false
}
```

jane-profile.json

The main difference between a JSON object and a regular JavaScript object – also called an object literal – comes down to the quotation marks. All the keys and string type values in a JSON object have to be wrapped in double quotation marks (`"`). JavaScript object literals are a bit more flexible. With object literals, you don't need to wrap keys and strings in double

quotation marks. Instead, you could use single quotation marks (`'`), or not use any type of quotation mark for the keys.
Here's what the code above might look like as a JavaScript object literal:

```
const profile = {
  name: 'Jane Doe',
  'favorite-game': 'Stardew Valley',
  subscriber: false
}
```

Note that the key `'favorite-game'` is wrapped in single quotes. With object literals, you'll need to wrap keys where the words are separated by dashes (-) in quotes.
If you'd like to avoid quotation marks, you could rewrite the key to use camel case (`favoriteGame`) or separate the words with an underscore (`favorite_game`) instead.

JSON arrays vs JavaScript arrays

JSON arrays work pretty much the same way as arrays in JavaScript, and can contain strings, booleans, numbers, and other JSON objects. For example:

```
[
  {
    "name": "Jane Doe",
    "favorite-game": "Stardew Valley",
    "subscriber": false
  },
  {
    "name": "John Doe",
    "favorite-game": "Dragon Quest XI",
    "subscriber": true
  }
]
```

profiles.json

Here's what that might look like in plain JavaScript:

```
const profiles = [
  {
    name: 'Jane Doe',
    'favorite-game': 'Stardew Valley',
    subscriber: false
  },
  {
    name: 'John Doe',
    'favorite-game': 'Dragon Quest XI',
    subscriber: true
  }
];
```

JSON as a string

You might be wondering, if there are JSON objects and arrays, couldn't you use it in your program like a regular JavaScript object literal or array?

The reason why you can't do this is that JSON is really just a string.

For example, when you write JSON in a separate file like with `jane-profile.json` or `profiles.json` above, that file actually contains text in the form of a JSON object or array, which happens to look like JavaScript.
And if you make a request to an API, it'll return something like this:

```
{"name":"Jane Doe","favorite-game":"Stardew Valley","subscriber":false}
```

Just like with text files, if you want to use JSON in your project, you'll need to parse or change it into something your programming language can understand. For instance, parsing a JSON object in Python will create a dictionary.

With that understanding, let's look at different ways to parse JSON in JavaScript.

How to parse JSON in the browser

If you're working with JSON in the browser, you're probably receiving or sending data through an API.

Let's take a look at a couple of examples.

How to parse JSON with `fetch`

The easiest way to get data from an API is with `fetch`, which includes the `.json()` method to parse JSON responses into a usable JavaScript object literal or array automagically.

Here's some code that uses `fetch` to make a `GET` request for a developer-themed joke from the free Chuck Norris Jokes API:

```
fetch('https://api.chucknorris.io/jokes/random?category=dev')
  .then(res => res.json())  // the .json() method parses the JSON response into a JS object literal
  .then(data => console.log(data));
```

If you run that code in the browser, you'll see something like this logged to the console:


```
{
    "categories": ["dev"],
    "created_at": "2020-01-05 13:42:19.324003",
    "icon_url": "https://assets.chucknorris.host/img/avatar/chuck-norris.png",
    "id": "elgv2wkvt8ioag6xywykbq",
    "updated_at": "2020-01-05 13:42:19.324003",
    "url": "https://api.chucknorris.io/jokes/elgv2wkvt8ioag6xywykbq",
    "value": "Chuck Norris's keyboard doesn't have a Ctrl key because nothing controls Chuck Norris."
}
```


While that looks like a JSON object, it's really a JavaScript object literal, and you can use it freely in your program.

How to stringify JSON with `JSON.stringify()`

But what if you want to send data to an API?

For instance, say you'd like to send a Chuck Norris joke to the Chuck Norris Jokes API so other people can read it later.

First, you'd write your joke as a JS object literal:

```
const newJoke = {
  categories: ['dev'],
  value: "Chuck Norris's keyboard is made up entirely of Cmd keys because Chuck Norris is always in command."
};
```

Then, since you're sending data to an API, you'd need to turn your `newJoke` object literal into a JSON string.

Fortunately, JavaScript includes a super helpful method to do just that – `JSON.stringify()`:

```
const newJoke = {
  categories: ['dev'],
  value: "Chuck Norris's keyboard is made up entirely of Cmd keys because Chuck Norris is always in command."
};

console.log(JSON.stringify(newJoke)); // {"categories":["dev"],"value":"Chuck Norris's keyboard is made up entirely of Cmd keys because Chuck Norris is always in command."}

console.log(typeof JSON.stringify(newJoke)); // string
```

While we're converting an object literal into a JSON string in this example, `JSON.stringify()` also works with arrays.

Finally, you'd just need to send your JSON stringified joke back to the API with a `POST` request.

Note that the Chuck Norris Jokes API doesn't actually have this feature. But if it did, here's what the code might look like:

```
const newJoke = {
  categories: ['dev'],
  value: "Chuck Norris's keyboard is made up entirely of Cmd keys because Chuck Norris is always in command."
};
```

```
fetch('https://api.chucknorris.io/jokes/submit', { // fake API endpoint
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify(newJoke), // turn the JS object literal into a JSON string
})
  .then(res => res.json())
  .then(data => console.log(data))
  .catch(err => {
    console.error(err);
  });
```

And just like that, you've parsed incoming JSON with `fetch` and used `JSON.stringify()` to convert a JS object literal into a JSON string.

How to work with local JSON files in the browser

Unfortunately, it's not possible (or advisable) to load a local JSON file in the browser.

`fetch` will throw an error if you try to load a local file. For example, say you have a JSON file with some jokes:


```
[
  {
    "categories": ["dev"],
    "created_at": "2020-01-05 13:42:19.324003",
    "icon_url": "https://assets.chucknorris.host/img/avatar/chuck-norris.png",
    "id": "elgv2wkvt8ioag6xywykbq",
    "updated_at": "2020-01-05 13:42:19.324003",
    "url": "https://api.chucknorris.io/jokes/elgv2wkvt8ioag6xywykbq",
    "value": "Chuck Norris's keyboard doesn't have a Ctrl key because nothing controls Chuck Norris."
  },
  {
    "categories": ["dev"],
    "created_at": "2020-01-05 13:42:19.324003",
    "icon_url": "https://assets.chucknorris.host/img/avatar/chuck-norris.png",
    "id": "ae-78cogr-cb6x9hluwqtw",
```

```
    "updated_at": "2020-01-05 13:42:19.324003",
    "url": "https://api.chucknorris.io/jokes/ae-78cogr-cb6x9hluwqtw",
    "value": "There is no Esc key on Chuck Norris' keyboard, because no one escapes Chuck Norris."
  }
]
```

jokes.json

And you want to parse it and create a list of jokes on a simple HTML page.

If you create a page with the following and open it in your browser:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
    <meta name="viewport" content="width=device-width" />
    <title>Fetch Local JSON</title>
  </head>
  <script>
  fetch("./jokes.json", { mode: "no-cors" }) // disable CORS because path does not contain http(s)
   .then((res) => res.json())
   .then((data) => console.log(data));
 </script>
</html>
```

index.html

You'll see this in the console:

```
Fetch API cannot load file://<path>/jokes.json. URL scheme "file" is not supported
```

By default, browsers don't allow access to local files for security reasons. This is a good thing, and you shouldn't try to work around this behavior.

Instead, the best thing to do is to convert the local JSON file into JavaScript. Fortunately, this is pretty easy since JSON syntax is so similar to JavaScript.

All you need to do is create a new file and declare your JSON as a variable:

```
const jokes = [
  {
    "categories": ["dev"],
    "created_at": "2020-01-05 13:42:19.324003",
    "icon_url": "https://assets.chucknorris.host/img/avatar/chuck-norris.png",
    "id": "elgv2wkvt8ioag6xywykbq",
    "updated_at": "2020-01-05 13:42:19.324003",
    "url": "https://api.chucknorris.io/jokes/elgv2wkvt8ioag6xywykbq",
    "value": "Chuck Norris's keyboard doesn't have a Ctrl key because nothing controls Chuck Norris."
  },
  {
    "categories": ["dev"],
    "created_at": "2020-01-05 13:42:19.324003",
    "icon_url": "https://assets.chucknorris.host/img/avatar/chuck-norris.png",
    "id": "ae-78cogr-cb6x9hluwqtw",
    "updated_at": "2020-01-05 13:42:19.324003",
    "url": "https://api.chucknorris.io/jokes/ae-78cogr-cb6x9hluwqtw",
    "value": "There is no Esc key on Chuck Norris' keyboard, because no one escapes Chuck Norris."
  }
]
```

jokes.js

And add it to your page as a separate script:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
    <meta name="viewport" content="width=device-width" />
```

```
    <title>Fetch Local JSON</title>
  </head>
  <script src="jokes.js"></script>
  <script>
  console.log(jokes);
 </script>
</html>
```

You'll be able to use the `jokes` array freely in your code. You could also use JavaScript modules to do the same thing,