BEN ALMAN

The content in this website is wildly out-of-date, but will be updated eventually. Until then, feel free to checkout my Github page as well as my latest musical project, The Entire Robot.

There's no such thing as a "JSON Object"

By "Cowboy" Ben Alman on March 3, 2010 2:54 PM

I want to clear up a common misconception. It's my belief that developers mistakenly call JavaScript Object literals "JSON Objects" because their syntax is identical to (well, a superset of) what is described in the JSON specification, but what the specification fails to mention explicitly is that since JSON is a "data-interchange language," *it's only actually JSON when it's used in a string context.*

Serialization and deserialization

When two applications/servers/languages/etc need to communicate, they tend to do so using strings, since strings can be interpreted in pretty much the same way in all languages. While a complex data structure is often described internally in terms of memory references, and represented with all kinds of curly, square, angle brackets or whitespace.. a string is just an ordered series of characters.

To this end, standard rules or syntaxes have been created to describe complex data structures as strings. JSON is one such syntax, as it can describe Objects, Arrays, Strings, Numbers, Booleans and null in a string context, which can be passed from application to application and decoded as needed. YAML and XML (and even request params) are two other popular data interchange (or serialization) formats, but of course we like JSON because we're JavaScript developers!

Literals

And just for reference, I'm going to quote Mozilla Developer Center here:

- Literals are fixed values, not variables, that you *literally* provide in your script. (Literals)
- A string literal is zero or more characters enclosed in double (") or single (') quotation marks. (String Literals)
- An object literal is a list of zero or more pairs of property names and associated values of an object, enclosed in curly braces ({}). (Object Literals)

So, when is JSON not JSON?

JSON was designed as a data interchange format, which happens to have a syntax that is a subset of JavaScript.

And as such, `{ "prop": "val" }` could be a JavaScript Object literal or a JSON string, depending in what context it's being used. If it's used in a string context (surrounded by single or double quotes, loaded from a text file, etc) it is a JSON string. If it's used in an Object literal context, it's an Object literal.

```
// This is a JSON String.
var foo = '{ "prop": "val" }';

// This is an Object literal.
var bar = { "prop": "val" };
```

Also, note that JSON has a very strict syntax, and while `{ "prop": "val" }` is valid JSON when used in a string context, `{ prop: "val" }` and `{ 'prop': 'val' }` are not valid JSON. All property names and string values must be surrounded by double quotes, not single quotes! And for you PHP guys, note that escaped single quotes are also invalid. Yes, Flickr, I'm talking to you. Either way, see the syntax rules for all of the details.

Putting things into context

So, all you smart-asses out there, you know who you are.. you're thinking "Well, isn't my JavaScript source code just one big string of text?"

Yes, of course it is. All your JavaScript and HTML (among other things) are, by themselves, strings of text.. until they are interpreted by the browser. That .js file or inline JavaScript code ceases to be "just one big string of text" the moment that the browser interprets it as JavaScript source… just like the page's innerHTML ceases to be "just one big string of text" as soon as it is converted into a DOM structure.

Again, it's all about context. Use that curly-brace-delimited JavaScript object in a string context and you've got a JSON string. Use it in an Object literal context, and you've got an Object literal.

The JSON object

So, maybe I fibbed a little. While Object literals are not "JSON Objects," there really is a JSON object, but it's something else entirely. In modern browsers, the JSON object is a native object with the static methods `JSON.parse` (deserialize a JSON string into an object) and `JSON.stringify`

(serialize an object into a JSON string). When you want to convert to and from JSON, you use these methods. In older browsers that don't provide a native JSON object, you can use json2.js to add this functionality.

And just in case you still don't understand, don't worry about it, here's a little cheat sheet:

```
// This is a JSON String, like what you'd get back from an AJAX request.
var my_json_string = '{ "prop": "val" }';

// This is how you deserialize that JSON String into an Object.
var my_obj = JSON.parse( my_json_string );

alert( my_obj.prop == 'val' ); // Alerts true, fancy that!

// And this is how you serialize an Object into a JSON String.
var my_other_json_string = JSON.stringify( my_obj );
```

Also, Paul Irish pointed out to me that Douglas Crockford uses the term "JSON object" in the JSON RFC, but in that context it means "Object as represented in a JSON string" and not "Object literal."

Extra credit

If you want to learn more about JSON, here are a few notable JSON-related links, for perusal at your leisure.

- JSON specification
- JSON RFC
- JSON on Wikipedia
- JSONLint - The JSON Validator
- JSON is not the same as JSON

Tags: json | object | object literal | string

Categories: Code | News

« Previous | Recent News | All News | Next »

POST A COMMENT

- Any of these HTML tags may be used for style: `a`, `b`, `i`, `br`, `p`, `strong`, `em`, `pre`, `code`.
- Multi-line JavaScript code should be wrapped in `<pre class="brush:js"></pre>`
  (supported syntax highlighting brushes: `js`, `css`, `php`, `plain`, `bash`, `ruby`, `html`, `xml`)
- Use `<` instead of `<` and `>` instead of `>` in the examples themselves.