Discovering JavaScript Object Notation

Charles Severance
University of Michigan

JSON is moving from being an underground secret, known and used by very few, to becoming the clear choice for mainstream data applications.

JavaScript Object Notation (JSON) is a popular format for data serialization. Programmers use it extensively to encode data for transfer between a server and an Ajax application, to connect two servers communicating via Web services, and in many other similar scenarios.

Yahoo’s Douglas Crockford is JSON’s self-appointed evangelist. He’s also involved in the JavaScript community, author of JavaScript: The Good Parts (Yahoo Press, 2008), and developer of JSLint, a tool that aims to improve the quality of JavaScript applications.

Crockford started using JSON in 2001, when he worked at State Software, a company where he and his colleagues built software demonstrating what would become the JavaScript/Ajax pattern. This pattern divided an application’s implementation between JavaScript running in the browser and a back-end server that housed the data:

I started with JavaScript, but my first application was facilitating communication between programs written in JavaScript and servers written in Java. I recognized that even though [JSON] was born out of JavaScript, it could be and should be language independent. I simplified it as much as possible, took some out, tried to make the simplest possible specification for how to structure

The earliest instance that I found of JavaScript being used as a data interchange format was at Netscape in 1996, so it’s an idea that has been around for a while. If you look at other data representations—like the property lists used at NeXt and then later at Apple—except for a couple of cosmetic changes, they [use] JSON notation as well. It seems like an inevitable representation for data—at least data that is intended to be consumed by programming languages, and that is all data.

The most common structures we use in programming are scalar variables, linear lists, and key-value pairs. JSON represents these structures in the most natural and direct serialization, greatly reducing the impedance mismatch between in-memory structures in applications and the serialization format. JSON is not only convenient but also efficient. Building JSON into JavaScript gives it a distinct advantage over other serialization formats such as XML when working with applications partially written in JavaScript.

Crockford started using JSON in 2001, when he worked at State Software, a company where he and his colleagues built software demonstrating what would become the JavaScript/Ajax pattern. This pattern divided an application’s implementation between JavaScript running in the browser and a back-end server that housed the data:

I started with JavaScript, but my first application was facilitating communication between programs written in JavaScript and servers written in Java. I recognized that even though [JSON] was born out of JavaScript, it could be and should be language independent. I simplified it as much as possible, took some out, tried to make the simplest possible specification for how to structure
data, and put it on the wire. And that ultimately became JSON.

In the beginning, JSON was more like a nice feature in JavaScript that happened to make writing these applications much simpler. But in 2001, XML was all the rage in data representation, and potential State Software customers were fearful of adopting anything other than XML. Crockford describes how a typical conversation with a potential customer would transpire:

(State Software) produced some brilliant demonstrations, and we were starting to make some progress to convince potential customers that they should adopt this style of application development. As part of the description we would say, We use this JSON idea for communicating data back and forth. They would say, JSON—what’s that? We would say, It was this thing we found in JavaScript that is really great, and then they would say, We can’t use that; we just committed to XML. Then we would say, XML is wrong for all these reasons: it’s inefficient and harder to use. Then they would say, We can’t use the thing you did because it’s not a standard. Then I would say, It’s a standard; it’s a proper subset of ECMA 262 (JavaScript), which is a standard. They would say, No, that’s not a standard.

The conversation with potential customers inevitably boiled down to the fact that they found it odd to use JavaScript notation for literal constants as a data interchange format. Crockford decided that to address this problem, the idea needed an acronym, a name, and a bit of “brand identity” so that people could talk about it and call it something other than “the literal/constant format for the JavaScript language.” As he tells it, I decided that if I wanted to be able to use this thing [JSON], I needed to make it a standard, so I bought json.org and put up a webpage and declared it as a standard. That’s it. That’s all I did. I didn’t go around trying to convince industry and government and everybody that this is what they should do. I just put up a one-page website. Over the years, people discovered it and realized it was so much easier [to use JSON].

JSON VERSUS XML

If you’ve worked as a programmer in the past decade, you’ve witnessed the constant background debate about whether XML or JSON is the right format for data representation. XML was always the well-understood “enterprise solution.” An extensive toolset was available for working with and manipulating XML, there were sophisticated mechanisms for precisely defining the XML schema, and developers could use extensive libraries to create and parse XML. Since the mid-1990s, XML seemed like the safe and obvious choice for data representation and interchange. Industry support for it was broad and well-organized, whereas JSON had one webpage and one man making presentations around the world about this cool idea.

So why, then, is JSON slowly displacing XML as the preferred way to serialize and exchange data? Why is the underdog winning?

When Ajax was formulated, the “X” in Ajax was supposed to be XML, and the smart kids right away realized this [XML] is too hard: we don’t want to be doing XML. Some of them discovered that they could use JSON instead, and that it was so much easier and so much faster. For a while, there was a debate [about whether to use XML or JSON in Ajax], but that didn’t last very long.

Even though JSON seems to be winning as the most natural data transfer representation when communicating from one program to another, developers continue to view XML (and HTML) as the best representation for document-style information such as word-processing files or webpages. Clearly, there isn’t a perfect, one-size-fits-all solution for data representation, but at least for now, JSON is dominating program-to-program data transfer.

THE FUTURE

Once programmers make the switch to JSON, they seldom switch back to XML for their program-to-program data serialization needs. But inevitably, as more programmers
This ability to add value to JSON without altering it is an essential feature, according to Crockford: 

"Probably the boldest design decision I made was to not put a version number on JSON so there is no mechanism for revising it. We are stuck with JSON: whatever it is in its current form, that's it. And that turns out to be its best feature. Because it is a basic infrastructure—it's the thing that you pile everything else on—it's equivalent to an alphabet in a language. We might make up lots of words and lots of ways of creating sentences, but it's very uncommon to make new letters. And that's the place where JSON lives, so it's good that it's not going to change.

Another effort that adds value to JSON without changing it is "JSON for Linked Data" (www.json-ld.org), which adds simple extensions to JSON so that it can easily represent large linked data structures typically represented via RDF. Like the JSON Media Type, JSON-LD is an extension that works without changing the JSON structure. In a sense, JSON-LD is just another language using JSON as its alphabet.

JSON is moving from being an underground secret, known and used by very few, to becoming the clear choice for mainstream data applications. But perhaps what's most exciting is the value-add capabilities that will become part of the developer's toolkit as JSON increasingly takes over the heavy lifting of data encoding and interchange.

Charles Severance, editor of the Computing Conversations column and Computer's multimedia editor, is a clinical associate professor and teaches in the School of Information at the University of Michigan. Contact him at csev@umich.edu. You can follow Severance on Twitter @drchuck.