

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- * A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.

- * An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

```
var myJSONObject = {"bindings": [  
    {"ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.*"},  
    {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex": "^delete.*"},  
    {"ircEvent": "PRIVMSG", "method": "randomURI", "regex": "^random.*"}  
    ]  
};
```

In this example, an object is created containing a single member "bindings", which contains an array containing three objects, each containing "ircEvent", "method", and "regex" members.

Members can be retrieved using dot or subscript operators.

```
myJSONObject.bindings[0].method // "newURI"
```

To convert a JSON text into an object, you can use the `eval()` function. `eval()` invokes the JavaScript compiler. Since JSON is a proper subset of JavaScript, the compiler will correctly

parse the text and produce an object structure. The text must be wrapped in parens to avoid tripping on an ambiguity in JavaScript's syntax.

```
var myObject = eval('(' + myJSONtext + ');');
```

The eval function is very fast. However, it can compile and execute any JavaScript program, so there can be security issues. The use of eval is indicated when the source is trusted and competent. It is much safer to use a JSON parser. In web applications over XMLHttpRequest, communication is permitted only to the same origin that provide that page, so it is trusted. But it might not be competent. If the server is not rigorous in its JSON encoding, or if it does not scrupulously validate all of its inputs, then it could deliver invalid JSON text that could be carrying dangerous script. The eval function would execute the script, unleashing its malice.

JSON Schema

There are several ways to verify the structure and data types inside a JSON object, much like an [XML](#) schema. JSON Schema is a specification for a JSON-based format for defining the structure of JSON data. JSON Schema provides a contract for what JSON data is required for a given application and how it can be modified, much like what XML Schema provides for XML. JSON Schema is intended to provide validation, documentation, and interaction control of JSON data. JSON Schema is based on the concepts from XML Schema, RelaxNG, and Kwalify, but is intended to be JSON-based, so that JSON data in the form of a schema can be used to validate JSON data, the same serialization/deserialization tools can be used for the schema and data, and it can be self descriptive. [Json.Com. "JSON Schema Proposal"](#).

Security issues

Although JSON is intended as a data serialization format, its design as a subset of the JavaScript programming language poses several security concerns. These concerns center on the use of a JavaScript interpreter to dynamically execute JSON text as JavaScript, thus exposing a program to errant or malicious script contained therein -- often a chief concern when dealing with data retrieved from the internet. While not the only way to process JSON, it is an easy and popular technique, stemming from JSON's design to be compatible with JavaScript's eval() function.

