chem-bla-ics

Programming in the Life Sciences #10: JavaScript Object Notation (JSON)

Egon Willighagen

Published October 30, 2013

Citation

Willighagen, E. (2013, October 30). Programming in the Life Sciences #10: JavaScript Object Notation (JSON). *Chem-bla-ics*. https://doi.org/10.59350/xdnrb-rrc91

Keywords

Pra3006, Json

As said, JSON is the format we will use as serialization format for answers given by the Open PHACTS LDA. The API actually supports XML, RDF, HTML, and TSV too, but I think JSON is a good balance between expressiveness and compactness. Moreover, and perhaps a much better argument, JSON works very well in a JavaScript environment: it is very easy to convert the serialization into a data model:

```
var jsonData = JSON.parse(jsonString);
```

Now, we previously covered maps. Maps have keys and values: the keys unlock a particular value. For example, take this JavaScript:

```
var map = { "key": "value", "key2": "value2" };
```

We define here a key-value object, and we can access the two values with the two keys:

```
map["key2"]; // == value2
```

These examples are JavaScript source code. Not a string. The content of the map variable is a data structure. But when we communicate with a web service, we need a (string) serialization of the data model, because we cannot send around memory pointers (which a variable is) because they are only valid on a single machine.

This is where the JSON format comes in. We can convert the content of the above map variable into a string representation with this code:

```
var mapStringified = JSON.stringify(map);
```

which gives us the following output:

```
{"key":"value","key2":"value2"}
```

This string looks an awful lot like the JavaScript code we wrote earlier.

And, likewise we can convert the JSON string back into a JavaScript data model again, with:

```
var mapAgain = JSON.parse(mapStringified);
```

Now, I did warn you earlier that values can be lists and maps itself again, so consider this JSON example from Wikipedia:

```
{
    "id": 1,
    "name": "Foo",
    "price": 123,
    "tags": [ "Bar", "Eek" ],
    "stock": {
        "warehouse": 300,
        "retail": 20
```

```
    }
}
```

Here we see that the value behind the stock key is another map, and the value behind the tags key is a list. This creates a quite flexible serialization format, which is happily used by Open PHACTS. (And for the semantic web readers, yes, we can make JSON more semantic. The Open PHACTS LDA supports a “rdfjson” format.)