JSON4MAT

Table of Contents

Jonas Almeida, April 2010

This manual was generated automatically by running pub.m .

JSON strings into (json2mat) and from (mat2json) Matlab structures. The name, "...4MAT", is a wink to an old, outdated but similalry minded XML toolbox XML4MAT where loose typing was argued for as being closer to Matlab's own loose assignment of types within a mat structure (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.84.9787)

This is the first version of JSON4MAT was developed specifically for COUCH4MAT (http://couch4mat.googlecode.com), a tbox developed to help with interoperability with CouchDB. A reflection of the idea of loose typing and dimensionality natural to Matlab environments is ported to the reading and writting of numerical matrices into JSON.

JSON2MAT

JavaScript Object Notation (JSON) has a much looser and smaller set of data types that Matlab. Also, unlike XML it does not explicitly make room for attributes to be defined to additionally characterize a data element such as size and class. As a consequence the JSON strings can be interpreted just as dynamically:

```
json='{lele:2,lili:4,lolo:[1,2,{lulu:5,bubu:[[1,2],[3,4],[5,6]]}]}';
mat=json2mat(json)
```

```
mat =

    lele: 2
    lili: 4
    lolo: {[1]  [2]  [1x1 struct]}
```

Dimensionality in numerical arrays

The array of pairs of numbers suggests a 3x2 matrix and indeed JSON2MAT made that conversion:

```
mat.lolo{3}.bubu
```

```
ans =

     1     2
     3     4
     5     6
```

Non-numerial arrays

When there are non-numerical elements in a JSON array the array of arrays will be directly converted into a Matlab cell of cells. Notice the last numerical element of bubu is now "a".

```
json2mat('{lele:2,lili:4,lolo:[1,2,{lulu:5,bubu:[[1,2],[3,4],[5,"a"]]}]}');
ans.lolo{3}
```

```
ans =

    lulu: 5
    bubu: {[1 2]  [3 4]  {1x2 cell}}
```

Unmatched dimensionality

The interpretation of arrays of arrays of numbers as corresponding to embedded dimensionality is performed by a try / catch command. Therefore, if teh dimensions don't match the cells of cells will be returned instead. In a future version maybe dimensionality in cell arrays will be coded in JSON2MAT as well, but presently that is only attempted for numerical arrays.

```
ans=json2mat('{lele:2,lili:4,lolo:[1,2,{lulu:5,bubu:[[1,2],[3,4],[5,6,7]]}]}');
ans.lolo{3}
```

```
ans =

    lulu: 5
    bubu: [1 2 3 4 5 6 7]
```

MAT2JSON

Since the original motivation for this toolbox was to read entries from a CouchDB deployment, the conversion of mat into json was designed with a single criterion in mind: that the reverse operation JSON2MAT would return the original mat structure. Accordingly, if we return to the mat structure generated from the json string produced under JSON2MAT above, one can confirm that the same string is returned from mat:

```
mat2json(mat)
```

```
ans =
```

```
{"lele":2,"lili":4,"lolo":[1,2,{"lulu":5,"bubu":[[1,2],[3,4],[5,6]]}]}
```

Non-standard convinience

There are some additional simplifications of JSON that are supported by this parser and may come handy even if they are not supported by standard js engines. For example the type of a=1 and b="1" is recognized as it should. It is less clear what to do with

```
{a:1,b:"1",c:d,d:[1 2 3],e:a b,f:1 2,g:["1",1,1E2]}
```

even if it pretty clear that d should have been noted as "d". This gets a bit trickier for d, e, f and g but you can confirm that this parser tries to treat it as numeric and if it doesn't work it assumes you forgot the ""

```
json2mat('{a:1,b:"1",c:d,d:[1 2 3],e:a b,f:1 2,g:["1",1,1E2]}')
```

```
ans =

    a: 1
    b: '1'
    c: 'd'
    d: [1 2 3]
    e: 'a b'
    f: '1 2'
    g: {'1'  [1]  [100]}
```

Published with MATLAB® 7.10