===============================================================================

= JSONLab =

= An open-source MATLAB/Octave JSON encoder and decoder =

===============================================================================

\*Copyright (C) 2011-2015 Qianqian Fang <fangq at nmr.mgh.harvard.edu>

\*License: BSD License, see License\_BSD.txt for details

\*Version: 1.0 (Optimus - Final)

-------------------------------------------------------------------------------

Table of Content:

I. Introduction

II. Installation

III.Using JSONLab

IV. Known Issues and TODOs

V. Contribution and feedback

-------------------------------------------------------------------------------

I. Introduction

JSON ([http://www.json.org/ JavaScript Object Notation]) is a highly portable,

human-readable and "[http://en.wikipedia.org/wiki/JSON fat-free]" text format

to represent complex and hierarchical data. It is as powerful as

[http://en.wikipedia.org/wiki/XML XML], but less verbose. JSON format is widely

used for data-exchange in applications, and is essential for the wild success

of [http://en.wikipedia.org/wiki/Ajax\_(programming) Ajax] and

[http://en.wikipedia.org/wiki/Web\_2.0 Web2.0].

UBJSON (Universal Binary JSON) is a binary JSON format, specifically

optimized for compact file size and better performance while keeping

the semantics as simple as the text-based JSON format. Using the UBJSON

format allows to wrap complex binary data in a flexible and extensible

structure, making it possible to process complex and large dataset

without accuracy loss due to text conversions.

We envision that both JSON and its binary version will serve as part of

the mainstream data-exchange formats for scientific research in the future.

It will provide the flexibility and generality achieved by other popular

general-purpose file specifications, such as

[http://www.hdfgroup.org/HDF5/whatishdf5.html HDF5], with significantly

reduced complexity and enhanced performance.

JSONLab is a free and open-source implementation of a JSON/UBJSON encoder

and a decoder in the native MATLAB language. It can be used to convert a MATLAB

data structure (array, struct, cell, struct array and cell array) into

JSON/UBJSON formatted strings, or to decode a JSON/UBJSON file into MATLAB

data structure. JSONLab supports both MATLAB and

[http://www.gnu.org/software/octave/ GNU Octave] (a free MATLAB clone).

-------------------------------------------------------------------------------

II. Installation

The installation of JSONLab is no different than any other simple

MATLAB toolbox. You only need to download/unzip the JSONLab package

to a folder, and add the folder's path to MATLAB/Octave's path list

by using the following command:

addpath('/path/to/jsonlab');

If you want to add this path permanently, you need to type "pathtool",

browse to the jsonlab root folder and add to the list, then click "Save".

Then, run "rehash" in MATLAB, and type "which loadjson", if you see an

output, that means JSONLab is installed for MATLAB/Octave.

-------------------------------------------------------------------------------

III.Using JSONLab

JSONLab provides two functions, loadjson.m -- a MATLAB->JSON decoder,

and savejson.m -- a MATLAB->JSON encoder, for the text-based JSON, and

two equivallent functions -- loadubjson and saveubjson for the binary

JSON. The detailed help info for the four functions can be found below:

=== loadjson.m ===

<pre>

data=loadjson(fname,opt)

or

data=loadjson(fname,'param1',value1,'param2',value2,...)

parse a JSON (JavaScript Object Notation) file or string

authors:Qianqian Fang (fangq<at> nmr.mgh.harvard.edu)

created on 2011/09/09, including previous works from

Nedialko Krouchev: http://www.mathworks.com/matlabcentral/fileexchange/25713

created on 2009/11/02

François Glineur: http://www.mathworks.com/matlabcentral/fileexchange/23393

created on 2009/03/22

Joel Feenstra:

http://www.mathworks.com/matlabcentral/fileexchange/20565

created on 2008/07/03

$Id: loadjson.m 452 2014-11-22 16:43:33Z fangq $

input:

fname: input file name, if fname contains "{}" or "[]", fname

will be interpreted as a JSON string

opt: a struct to store parsing options, opt can be replaced by

a list of ('param',value) pairs - the param string is equivallent

to a field in opt. opt can have the following

fields (first in [.|.] is the default)

opt.SimplifyCell [0|1]: if set to 1, loadjson will call cell2mat

for each element of the JSON data, and group

arrays based on the cell2mat rules.

opt.FastArrayParser [1|0 or integer]: if set to 1, use a

speed-optimized array parser when loading an

array object. The fast array parser may

collapse block arrays into a single large

array similar to rules defined in cell2mat; 0 to

use a legacy parser; if set to a larger-than-1

value, this option will specify the minimum

dimension to enable the fast array parser. For

example, if the input is a 3D array, setting

FastArrayParser to 1 will return a 3D array;

setting to 2 will return a cell array of 2D

arrays; setting to 3 will return to a 2D cell

array of 1D vectors; setting to 4 will return a

3D cell array.

opt.ShowProgress [0|1]: if set to 1, loadjson displays a progress bar.

output:

dat: a cell array, where {...} blocks are converted into cell arrays,

and [...] are converted to arrays

examples:

dat=loadjson('{"obj":{"string":"value","array":[1,2,3]}}')

dat=loadjson(['examples' filesep 'example1.json'])

dat=loadjson(['examples' filesep 'example1.json'],'SimplifyCell',1)

</pre>

=== savejson.m ===

<pre>

json=savejson(rootname,obj,filename)

or

json=savejson(rootname,obj,opt)

json=savejson(rootname,obj,'param1',value1,'param2',value2,...)

convert a MATLAB object (cell, struct or array) into a JSON (JavaScript

Object Notation) string

author: Qianqian Fang (fangq<at> nmr.mgh.harvard.edu)

created on 2011/09/09

$Id: savejson.m 458 2014-12-19 22:17:17Z fangq $

input:

rootname: the name of the root-object, when set to '', the root name

is ignored, however, when opt.ForceRootName is set to 1 (see below),

the MATLAB variable name will be used as the root name.

obj: a MATLAB object (array, cell, cell array, struct, struct array).

filename: a string for the file name to save the output JSON data.

opt: a struct for additional options, ignore to use default values.

opt can have the following fields (first in [.|.] is the default)

opt.FileName [''|string]: a file name to save the output JSON data

opt.FloatFormat ['%.10g'|string]: format to show each numeric element

of a 1D/2D array;

opt.ArrayIndent [1|0]: if 1, output explicit data array with

precedent indentation; if 0, no indentation

opt.ArrayToStruct[0|1]: when set to 0, savejson outputs 1D/2D

array in JSON array format; if sets to 1, an

array will be shown as a struct with fields

"\_ArrayType\_", "\_ArraySize\_" and "\_ArrayData\_"; for

sparse arrays, the non-zero elements will be

saved to \_ArrayData\_ field in triplet-format i.e.

(ix,iy,val) and "\_ArrayIsSparse\_" will be added

with a value of 1; for a complex array, the

\_ArrayData\_ array will include two columns

(4 for sparse) to record the real and imaginary

parts, and also "\_ArrayIsComplex\_":1 is added.

opt.ParseLogical [0|1]: if this is set to 1, logical array elem

will use true/false rather than 1/0.

opt.NoRowBracket [1|0]: if this is set to 1, arrays with a single

numerical element will be shown without a square

bracket, unless it is the root object; if 0, square

brackets are forced for any numerical arrays.

opt.ForceRootName [0|1]: when set to 1 and rootname is empty, savejson

will use the name of the passed obj variable as the

root object name; if obj is an expression and

does not have a name, 'root' will be used; if this

is set to 0 and rootname is empty, the root level

will be merged down to the lower level.

opt.Inf ['"$1\_Inf\_"'|string]: a customized regular expression pattern

to represent +/-Inf. The matched pattern is '([-+]\*)Inf'

and $1 represents the sign. For those who want to use

1e999 to represent Inf, they can set opt.Inf to '$11e999'

opt.NaN ['"\_NaN\_"'|string]: a customized regular expression pattern

to represent NaN

opt.JSONP [''|string]: to generate a JSONP output (JSON with padding),

for example, if opt.JSONP='foo', the JSON data is

wrapped inside a function call as 'foo(...);'

opt.UnpackHex [1|0]: conver the 0x[hex code] output by loadjson

back to the string form

opt.SaveBinary [0|1]: 1 - save the JSON file in binary mode; 0 - text mode.

opt.Compact [0|1]: 1- out compact JSON format (remove all newlines and tabs)

opt can be replaced by a list of ('param',value) pairs. The param

string is equivallent to a field in opt and is case sensitive.

output:

json: a string in the JSON format (see http://json.org)

examples:

jsonmesh=struct('MeshNode',[0 0 0;1 0 0;0 1 0;1 1 0;0 0 1;1 0 1;0 1 1;1 1 1],...

'MeshTetra',[1 2 4 8;1 3 4 8;1 2 6 8;1 5 6 8;1 5 7 8;1 3 7 8],...

'MeshTri',[1 2 4;1 2 6;1 3 4;1 3 7;1 5 6;1 5 7;...

2 8 4;2 8 6;3 8 4;3 8 7;5 8 6;5 8 7],...

'MeshCreator','FangQ','MeshTitle','T6 Cube',...

'SpecialData',[nan, inf, -inf]);

savejson('jmesh',jsonmesh)

savejson('',jsonmesh,'ArrayIndent',0,'FloatFormat','\t%.5g')

</pre>

=== loadubjson.m ===

<pre>

data=loadubjson(fname,opt)

or

data=loadubjson(fname,'param1',value1,'param2',value2,...)

parse a JSON (JavaScript Object Notation) file or string

authors:Qianqian Fang (fangq<at> nmr.mgh.harvard.edu)

created on 2013/08/01

$Id: loadubjson.m 436 2014-08-05 20:51:40Z fangq $

input:

fname: input file name, if fname contains "{}" or "[]", fname

will be interpreted as a UBJSON string

opt: a struct to store parsing options, opt can be replaced by

a list of ('param',value) pairs - the param string is equivallent

to a field in opt. opt can have the following

fields (first in [.|.] is the default)

opt.SimplifyCell [0|1]: if set to 1, loadubjson will call cell2mat

for each element of the JSON data, and group

arrays based on the cell2mat rules.

opt.IntEndian [B|L]: specify the endianness of the integer fields

in the UBJSON input data. B - Big-Endian format for

integers (as required in the UBJSON specification);

L - input integer fields are in Little-Endian order.

output:

dat: a cell array, where {...} blocks are converted into cell arrays,

and [...] are converted to arrays

examples:

obj=struct('string','value','array',[1 2 3]);

ubjdata=saveubjson('obj',obj);

dat=loadubjson(ubjdata)

dat=loadubjson(['examples' filesep 'example1.ubj'])

dat=loadubjson(['examples' filesep 'example1.ubj'],'SimplifyCell',1)

</pre>

=== saveubjson.m ===

<pre>

json=saveubjson(rootname,obj,filename)

or

json=saveubjson(rootname,obj,opt)

json=saveubjson(rootname,obj,'param1',value1,'param2',value2,...)

convert a MATLAB object (cell, struct or array) into a Universal

Binary JSON (UBJSON) binary string

author: Qianqian Fang (fangq<at> nmr.mgh.harvard.edu)

created on 2013/08/17

$Id: saveubjson.m 440 2014-09-17 19:59:45Z fangq $

input:

rootname: the name of the root-object, when set to '', the root name

is ignored, however, when opt.ForceRootName is set to 1 (see below),

the MATLAB variable name will be used as the root name.

obj: a MATLAB object (array, cell, cell array, struct, struct array)

filename: a string for the file name to save the output UBJSON data

opt: a struct for additional options, ignore to use default values.

opt can have the following fields (first in [.|.] is the default)

opt.FileName [''|string]: a file name to save the output JSON data

opt.ArrayToStruct[0|1]: when set to 0, saveubjson outputs 1D/2D

array in JSON array format; if sets to 1, an

array will be shown as a struct with fields

"\_ArrayType\_", "\_ArraySize\_" and "\_ArrayData\_"; for

sparse arrays, the non-zero elements will be

saved to \_ArrayData\_ field in triplet-format i.e.

(ix,iy,val) and "\_ArrayIsSparse\_" will be added

with a value of 1; for a complex array, the

\_ArrayData\_ array will include two columns

(4 for sparse) to record the real and imaginary

parts, and also "\_ArrayIsComplex\_":1 is added.

opt.ParseLogical [1|0]: if this is set to 1, logical array elem

will use true/false rather than 1/0.

opt.NoRowBracket [1|0]: if this is set to 1, arrays with a single

numerical element will be shown without a square

bracket, unless it is the root object; if 0, square

brackets are forced for any numerical arrays.

opt.ForceRootName [0|1]: when set to 1 and rootname is empty, saveubjson

will use the name of the passed obj variable as the

root object name; if obj is an expression and

does not have a name, 'root' will be used; if this

is set to 0 and rootname is empty, the root level

will be merged down to the lower level.

opt.JSONP [''|string]: to generate a JSONP output (JSON with padding),

for example, if opt.JSON='foo', the JSON data is

wrapped inside a function call as 'foo(...);'

opt.UnpackHex [1|0]: conver the 0x[hex code] output by loadjson

back to the string form

opt can be replaced by a list of ('param',value) pairs. The param

string is equivallent to a field in opt and is case sensitive.

output:

json: a binary string in the UBJSON format (see http://ubjson.org)

examples:

jsonmesh=struct('MeshNode',[0 0 0;1 0 0;0 1 0;1 1 0;0 0 1;1 0 1;0 1 1;1 1 1],...

'MeshTetra',[1 2 4 8;1 3 4 8;1 2 6 8;1 5 6 8;1 5 7 8;1 3 7 8],...

'MeshTri',[1 2 4;1 2 6;1 3 4;1 3 7;1 5 6;1 5 7;...

2 8 4;2 8 6;3 8 4;3 8 7;5 8 6;5 8 7],...

'MeshCreator','FangQ','MeshTitle','T6 Cube',...

'SpecialData',[nan, inf, -inf]);

saveubjson('jsonmesh',jsonmesh)

saveubjson('jsonmesh',jsonmesh,'meshdata.ubj')

</pre>

=== examples ===

Under the "examples" folder, you can find several scripts to demonstrate the

basic utilities of JSONLab. Running the "demo\_jsonlab\_basic.m" script, you

will see the conversions from MATLAB data structure to JSON text and backward.

In "jsonlab\_selftest.m", we load complex JSON files downloaded from the Internet

and validate the loadjson/savejson functions for regression testing purposes.

Similarly, a "demo\_ubjson\_basic.m" script is provided to test the saveubjson

and loadubjson pairs for various matlab data structures.

Please run these examples and understand how JSONLab works before you use

it to process your data.

-------------------------------------------------------------------------------

IV. Known Issues and TODOs

JSONLab has several known limitations. We are striving to make it more general

and robust. Hopefully in a few future releases, the limitations become less.

Here are the known issues:

# 3D or higher dimensional cell/struct-arrays will be converted to 2D arrays;

# When processing names containing multi-byte characters, Octave and MATLAB \

can give different field-names; you can use feature('DefaultCharacterSet','latin1') \

in MATLAB to get consistant results

# savejson can not handle class and dataset.

# saveubjson converts a logical array into a uint8 ([U]) array

# an unofficial N-D array count syntax is implemented in saveubjson. We are \

actively communicating with the UBJSON spec maintainer to investigate the \

possibility of making it upstream

# loadubjson can not parse all UBJSON Specification (Draft 9) compliant \

files, however, it can parse all UBJSON files produced by saveubjson.

-------------------------------------------------------------------------------

V. Contribution and feedback

JSONLab is an open-source project. This means you can not only use it and modify

it as you wish, but also you can contribute your changes back to JSONLab so

that everyone else can enjoy the improvement. For anyone who want to contribute,

please download JSONLab source code from it's subversion repository by using the

following command:

svn checkout svn://svn.code.sf.net/p/iso2mesh/code/trunk/jsonlab jsonlab

You can make changes to the files as needed. Once you are satisfied with your

changes, and ready to share it with others, please cd the root directory of

JSONLab, and type

svn diff > yourname\_featurename.patch

You then email the .patch file to JSONLab's maintainer, Qianqian Fang, at

the email address shown in the beginning of this file. Qianqian will review

the changes and commit it to the subversion if they are satisfactory.

We appreciate any suggestions and feedbacks from you. Please use iso2mesh's

mailing list to report any questions you may have with JSONLab:

http://groups.google.com/group/iso2mesh-users?hl=en&pli=1

(Subscription to the mailing list is needed in order to post messages).