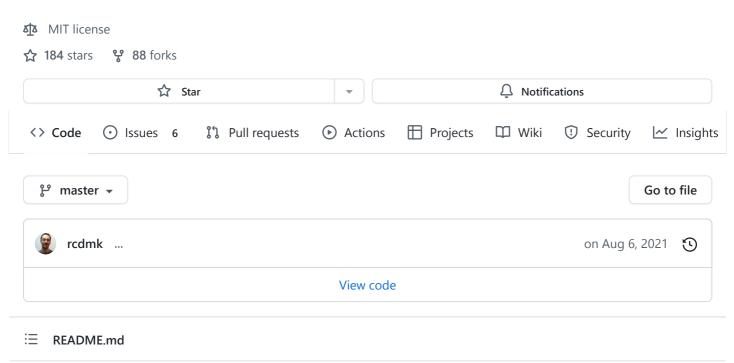


A fast classic ASP JSON parser and encoder for easy JSON manipulation to work with the new JavaScript MV* libraries and frameworks.



JSON object class 3.8.1

By RCDMK - rcdmk[at]hotmail[dot]com

Licence:

MIT license: http://opensource.org/licenses/mit-license.php

The MIT License (MIT)

Copyright (c) 2016 RCDMK - rcdmk[at]hotmail[dot]com

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

How to use:

This lib requires LCID (Locale Code IDentifier) property of your ASP application before use. You can do this by setting the LCID in one of the following ways:

- On the page declaration at the top of the page to set it to the entire page (eg.: <%@ LCID=1046 %>), OR
- On the Session object to set it to all pages in the entire session (eg.: Session.LCID = 1046), OR
- On the Response object, before using the class, to set it beyond this point on the page (eg.: Response.LCID = 1046)

```
Response.LCID = 1046 ' REQUIRED! Set your LCID here (1046 = Brazilian). Could also be the LCID p
' instantiate the class
set JSON = New JSONobject
' add properties
JSON.Add "prop1", "someString"
JSON.Add "prop2", 12.3
JSON.Add "prop3", Array(1, 2, "three")
' remove properties
JSON.Remove "prop2"
JSON.Remove "thisDoesNotExistsAndWillDoNothing"
' change some values
JSON.Change "prop1", "someOtherString"
JSON.Change "prop4", "thisWillBeCreated" ' this property doen't exists and will be created autom
' get the values
Response.Write JSON.Value("prop1") & "<br>"
Response.Write JSON.Value("prop2") & "<br>"
Response.Write JSON("prop3").Serialize() & "<br>" ' default function is equivalent to `.Value(pr
Response.Write JSON("prop4") & "<br>"
' get the JSON formatted output
Dim jsonString
jsonString = JSON.Serialize() ' this will contain the string representation of the JSON object
JSON.Write() ' this will write the output to the Response - equivalent to: Response.Write JSON.S
' load and parse some JSON formatted string
jsonString = "[{ ""strings"" : ""valorTexto"", ""numbers"": 123.456, ""arrays"": [1, ""2"", 3.4,
set oJSONoutput = JSON.Parse(jsonString) ' this method returns the parsed object. Arrays are par
                        'outputs: '{"data":[{"strings":"valorTexto","numbers":123.456,"arrays":
JSON.Write()
oJSONoutput.Write() ' outputs: '[{"strings":"valorTexto", "numbers":123.456, "arrays":[1,"2",3.4,[
' if the string represents an object (not an array of objects), the current object is returned s
jsonString = "{ ""strings"" : ""valorTexto"", ""numbers"": 123.456, ""arrays"": [1, ""2"", 3.4,
```

```
JSON.Parse(JsonString)
JSON.Write() ' outputs: '{"strings":"valorTexto","numbers":123.456,"arrays":[1,"2",3.4,[5,6,[7,8]]
```

To load records from a database:

```
' load records from an ADODB.Recordset
dim cn, rs
set cn = CreateObject("ADODB.Connection")
cn.Open "yourConnectionStringGoesHere"
set rs = cn.execute("SELECT id, nome, valor FROM pedidos ORDER BY id ASC")
' this could also be:
' set rs = CreateObject("ADODB.Recordset")
'rs.Open "SELECT id, nome, valor FROM pedidos ORDER BY id ASC", cn
JSON.LoadRecordset rs
JSONarr.LoadRecordset rs
rs.Close
cn.Close
set rs = Nothing
set cn = Nothing
JSON.Write()
                        ' outputs: {"data":[{"id":1,"nome":"nome 1","valor":10.99},{"id":2,"nome
                        ' outputs: [{"id":1,"nome":"nome 1","valor":10.99},{"id":2,"nome":"nome
JSONarr.Write()
```

To change the default property name ("data") when loading arrays and recordsets, use the defaultPropertyName property:

If you want to use arrays, I have something for you too

To loop arrays you have to access the items property of the JSONarray object and you can also access the items trough its index:

```
dim i, item
```

```
' more readable loop
for each item in JSONarr.items
        if isObject(item) and typeName(item) = "JSONobject" then
                item.write()
        else
                response.write item
        end if
        response.write "<br>"
next
' faster but less readable
for i = 0 to JSONarr.length - 1
        if isObject(JSONarr(i)) then
                set item = JSONarr(i)
                if typeName(item) = "JSONobject" then
                        item.write()
                else
                        response.write item
                end if
        else
                item = JSONarr(i)
                response.write item
        end if
        response.write "<br>"
next
```

Releases

28 tags

Packages

No packages published

Contributors 8

















Languages

Classic ASP 100.0%