



*I Wish I Knew How To ...*

*Begin Programming*

*JSON with Xojo*

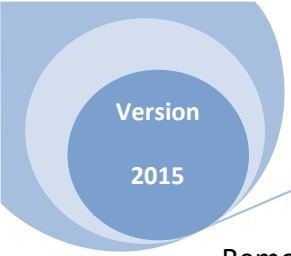
*November 2015 Edition (1.1)*

By Eugene Dakin

## Table of Contents

Chapter 1 - Introduction to Programming with JSON.....	8
JSON is not a Database.....	8
Why JSON .....	8
Data Types.....	9
Number.....	9
String.....	9
Boolean.....	9
Array .....	10
Object .....	10
WhiteSpace.....	10
Null.....	10
Human Readable .....	10
Free Online JSON Validator/Parser .....	11
JSON Rules.....	12
JSON Concepts.....	12
Old Vs New Framework.....	14
Chapter 2 – Classic Reading .....	15
Loading JSON Data .....	15
Count JSON Items.....	17
Show JSON Data .....	19
Children .....	22
Numbers.....	24
Boolean.....	27
Parent Array .....	30
Child Array.....	34
Manual Walking .....	38
Iterate Walking.....	41
Iterate Numbers .....	45
Iterate Booleans.....	50
Iterate Object and Array .....	54
Single Nested Data .....	59
Chapter 3 – Classic Writing .....	62
Parent JSON.....	62
Child and GrandChild JSON .....	64
Multiple JSONItems.....	67
Add Numbers .....	70
Add Boolean .....	73
Parent Array .....	76

Child Array .....	79
Insert Child Array.....	82
Chapter 4 – Classic Dictionary.....	86
Create .....	86
Count.....	88
Remove.....	89
Lookup.....	91
Clear .....	93
HasKey .....	94
Array .....	96
JSONItem to Dictionary .....	98
Dictionary to JSONItem .....	98
Chapter 5 – Remove.....	100
Remove Parent Data .....	100
Remove Child Data .....	105
Remove Array Data .....	109
Chapter 6 – New Framework Reading .....	114
Illegal Cast Exception.....	114
Text vs String .....	114
Using.....	115
Load and Show JSON Data .....	115
Children .....	118
GrandChildren .....	121
Parent Array .....	124
Children Array .....	127
Children Walk .....	130
Number and Boolean Walk .....	135
Single Nested Data .....	140
Chapter 7 – New Framework Writing .....	143
Parent Object .....	143
Child Objects .....	146
Numerical and Boolean.....	149
Parent Array .....	152
Child Array.....	155
Chapter 8 – New Framework Dictionary .....	158
Create .....	158
Count.....	160
Remove.....	161



Version

2015

Remove All.....	163
Lookup.....	165
Haskey .....	167
Array.....	168
Clone.....	169
Chapter 9 – Stock Quotes .....	170
Index.....	176

If the user successfully selected a file then the JSONFile Exists argument will be true. Loading a JSON file has the same steps as loading a text file, a `TextInputStream` variable is created and a string variable (`JSONData`) is created to hold the string data from the file.

A try-catch statement is used to try and execute code, and if there is a problem (error) then the catch portion of the statement is executed. `TextInputStream` calls the open method for the JSONfile and then places the raw string data into the `JSONStream` variable. All of the data is read from the `JSONStream` and is placed in the string `JSONData`. Larger files may take longer to load, which is why loading with a stream of data is important.

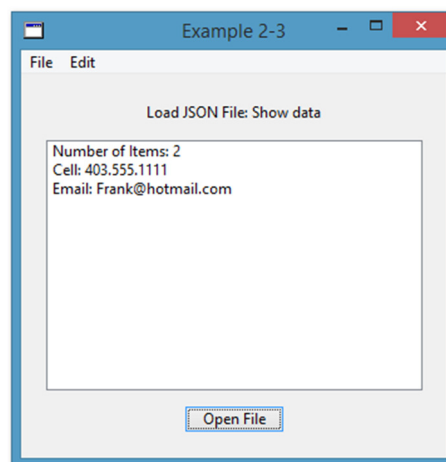
All rows in `Listbox1` are deleted which removes previous data if `Listbox1` was pressed twice. The JSON item uses the `Count` function to provide the number of JSON Name:Value pairs and the `ToText` portion converts the data to a text format which is viewed in `Listbox1`.

This example shows how to count the number of Name:Value pairs and show the result in a `Listbox`.

### ***Show JSON Data***

This example shows how to display the Name:Value pair in a listbox with Xojo. The sample file contains only Parent data without Child or GrandChild data, meaning there is only one node. Opening the JSON file `SimpleList` will give the following information in Example 2-3.

**Figure 5. Example 2-3: Showing JSON Data in a Listbox**



Using code from the previous example, the pushbutton action event opens a folder item for the user to choose the JSON file. When the JSON file is selected and opened, the number of items in the parent node are counted (2) and the two Name:Value pairs are shown in `Listbox1`.

### Code 9. SimpleList.JSON File

```
{  
  "Cell": "403.555.1111",  
  "Email": "Frank@hotmail.com"  
}
```

There are two Name:Value pairs that are separated by a comma and all the data is surrounded by two curly brackets.

Xojo code to read the node is shown below.

### Code 10. Example 2-3: Showing Parent Node Information

```
//Holds JSON data  
Dim JSONData as String  
  
//Get JSON File  
Dim JSONFile as FolderItem  
JSONFile = GetOpenFolderItem("")  
  
'Check to see if the file exists  
If JSONFile.Exists Then  
  'Try to load the file  
  Dim JSONStream as TextInputStream  
  Try //loading the file data  
    JSONStream = TextInputStream.Open(JSONFile)  
    JSONData = JSONStream.ReadAll  
    JSONStream.Close  
  Catch e as JSONException  
    MsgBox("Error: " + e.Message)  
    Return  
  End  
Else  
  MsgBox("JSON file does not exist")  
  Return  
End If
```

```
//Load the JSON data
Dim js as New JSONItem(JSONData)
//Show number of items
Listbox1.AddRow "Number of Items: " + js.Count.ToText

//Show JSON Name/Value pairs
Dim JSONName, JSONValue as String
Dim i as Integer
For i = 0 to js.Count-1
    JSONName = js.Name(i) //Get Name
    JSONValue = js.Value(JSONName) //Get Value
    Listbox1.AddRow JSONName + ": " + JSONValue //Show both
Next
```

The first few lines of code create a variable (JSONFile) from a FolderItem. The GetFolderItem command opens the folder item so the user can choose and select a file. In this example find and select the SimpleList JSON file that is in the Chapter 2 folder.

If the user successfully selected a file then the JSONFile Exists argument will be true. Loading a JSON file has the same steps as loading a text file, a TextInputStream variable is created and a string variable (JSONData) is created to hold the string data from the file.

A try-catch statement is used to try and execute code, and if there is a problem (error) then the catch portion of the statement is executed. TextInputStream calls the open method for the JSONfile and then places the raw string data into the JSONStream variable. All of the data is read from the JSONStream and is placed in the string JSONData. Larger files may take longer to load, which is why loading with a stream of data is important.

All rows in Listbox1 are deleted which removes previous data if Listbox1 was pressed twice. The JSON item uses the Count function to provide the number of JSON Name:Value pairs and the ToText portion converts the data to a text format which is viewed in Listbox1.

Two string variables are created to hold the name and value of each data pair. The name is first required because retrieving the value requires the name. A for-next loop is created from zero to the number of parent items with the Count function. The first name in the array (js.Name(i)) is passed to the JSONName string. Using the JSONName string, the value (js.Value(JSONName)) is then placed in the string variable JSONValue. A row is added to the listbox with the name (JSONName) and the value of the name (JSONValue). The for-next loop continues until the last JSON node has been read and the Name:Value data pair shown in the listbox.

This example shows how to read and show the data from a single JSON node which has two data pairs.

# Index

- AddRow, 21
- Append, 77, 83
- Array - Child, 34
- Array - Parent, 30
- Auto, 124
- Boolean Add, 73
- BooleanValue, 29
- Catch, 16
- Child, 29, 60
- Concepts, 12
- Count, 18, 32, 161
- Create, 63
- Data Types, 9
- DeleteAllRows, 18
- Dictionary, 86, 117
  - Array, 96
  - Clear, 93
  - Count, 89, 161
  - HasKey, 95, 168
  - Key, 87
  - Lookup, 92, 166
  - New, 159
  - Remove, 90, 162
  - RemoveAll, 163
  - Value, 87, 159
- DictionaryEntry, 117, 161
  - Key, 161
  - Value, 161
- DictionaryEntry, 134
- Error
  - IllegalCastException, 114
  - String, 114
  - Type Text, 114
- Examples
  - 02-01 Load JSON Data, 16
  - 02-02 Count JSON Items, 18
  - 02-03 Show JSON Node Data, 20
  - 02-04 View Child JSON, 23
  - 02-05 View JSON Child Number, 25
  - 02-06 View JSON Boolean, 28
  - 02-07 Parse Parent Array, 31
  - 02-08 Parse Child Array, 35
  - 02-09 Manual Walking JSON Data, 39
  - 02-10 Iterate Walking JSONItem, 44
  - 02-11 Iterate Walking Number JSONItem, 49
  - 02-12 Iterate Walking Boolean JSONItem, 53
  - 02-13 Iterate Mixed Object and Array, 57
  - 02-14 Single Key
    - Value Pair Reading, 60
  - 03-01 Create JSON Parent, 63
  - 03-02 Create JSON Child and GrandChildren, 65
  - 03-03 Create Multiple JSON Items, 68
  - 03-04 Create JSON Numbers, 71
  - 03-05 Create JSON Boolean Value, 74
  - 03-06 Create Parent Array, 77
  - 03-07 Create Child Array, 80
  - 03-08 Create Child Array Insert, 83
  - 04-01 Create Classic Dictionary, 86
  - 04-02 Dictionary Count, 88



- 04-03 Dictionary Remove, 90
- 04-04 Dictionary Lookup, 92
- 04-05 Dictionary Clear, 93
- 04-06 Dictionary HasKey, 95
- 04-07 Dictionary Array, 96
- 04-08 Dictionary to JSONItem, 99
- 05-01 Remove Parent Object Data, 101
- 05-02 Remove Child Object Data, 106
- 05-03 Remove Child array Data, 110
- 06-01 Load and Show JSON, 116
- 06-02 Read Child Key, 119
- 06-03 Show JSON GrandChild Object, 122
- 06-04 JSON Parent Array, 125
- 06-05 JSON Child Array, 128
- 06-06 Walk Child Nodes, 134
- 06-07 Walk Child Nodes Boolean and Numerical, 138
- 06-08 Retrieve Single JSON Child Value, 141
- 07-01 Create Parent JSON, 144
- 07-02 Create Child JSON, 147
- 07-03 Create Child Numerical and Boolean Data, 150
- 07-04 Create Parent Array JSON, 153
- 07-05 Create Child JSON Array, 156
- 08-01 Create New Framework Dictionary, 159
- 08-02 Count Dictionary Entries, 160
- 08-03 Remove One Dictionary Entry, 162
- 08-04 Remove All Dictionary Entries, 163
- 08-06 Dictionary HasKey, 167
- 08-07 Clone Dictionary, 169
- 09-01 Parse JSON Stock Data, 173
- Exists, 16
- Filename
  - ajax, 8
  - JSON, 8
  - txt, 8
- FolderItem, 16
- For-Each, 134
- For-Next, 89
- Framework, 14
- GenerateJSON, 145
- GetOpenFolderItem, 16
- HasKey, 168
- HasName, 102
- HTTPSocket, 173
- IllegalCastException, 114
- Insert, 84
- Introduction, 8
- Introspection GetType, 134
- Introspection GetType Name, 45, 136
  - Boolean, 45, 136
  - Byte, 45, 136
  - Double, 45, 136
  - Int32, 45, 136
  - Int64, 45, 136
  - Int8, 45, 136
  - Integer, 45, 136
  - Short, 45, 136
  - String, 45, 136
  - Text, 45, 136
  - UInt16, 45, 136
  - UInt32, 45, 136
  - UInt64, 45, 136
  - UInt8, 45, 136
  - Variant, 45, 136
- isArray, 32, 35
- JavaScript Object Notation, 8
- JSON, 8

JSON Validator, 11

JSONException, 16, 35

JSONItem, 14

- Append, 77, 83
- BooleanValue, 29
- Child, 29
- HasName, 102
- Insert, 84
- IsArray, 35
- Name, 32
- New, 18
- Remove, 102
- Value, 29, 36

Listbox

- ColumnCount, 88

Lookup, 166

Name, 21, 32, 60

NameSpace, 115

New, 18

New Framework, 14

Number Add, 70

Old Framework, 14

Open, 16

ParseJSON, 117

Parser, 11

ReadAll, 16

Recursion, 133, 138

Remove, 102, 162

RemoveAll, 163

Rules, 12

Socket, 173

Socket.Get, 173

String, 114

String Type

- ", 9
- /, 9
- b, 9
- f, 9
- n, 9
- r, 9
- t, 9
- u, 9

String Type:, 9

Text, 114

TextInputStream, 16

TextOutputStream, 63

Title, 68

ToString, 63

Try-Catch, 16

Typography, 11

Using, 115

Validator, 11

Value, 21, 29, 36

WhiteSpace, 11

Write, 63

Xojo Core DictionaryEntry, 117

See DictionaryEntry, 117  
Xojo Data ParseJSON, 117

Xojo.Core, 14

Xojo.Core.Dictionary

See Dictionary, 159  
Xojo.Data, 14

xojo.Data.GenerateJSON, 145

The 'I Wish I Knew' series contains technical data and advice that makes sense and contains practical and numerous examples with explanations to allow you to ease into the steep programming curve. You can extend Xojo applications today!

This book "I Wish I Knew How to ... Begin Programming JSON with Xojo" shows you how to create, parse, and read JSON data. There is much JSON data that is free, and for a fee there is a very large repository of information. This book is more than a cheat sheet, it provides an introduction to the basics of working with JSON in Xojo.

The book is written as a guide and reference to Xojo programmers who program Desktop Applications in Windows, Mac, or Linux.

There are 8 chapters and contains over 170 pages with over 50 example programs. Examples were tested with Xojo 2015 r2.4 and Windows 8.1, 10, OSX Yosemite 10.10.4, and Ubuntu 15.04-32bit.

Examples include parsing, creating, walking through children, arrays and more. Both Classic and New Framework examples are provided. Many screenshots have been added to show the results of the code with an index to help find topics quickly.

This is one of many books at Great White Software. This book can be purchased at <http://great-white-software.com/rblibrary/> where many great Xojo resources are available.

Happy programming!

Eugene

---

**Eugene Dakin MBA, Ph.D., P.Chem.**, is an author of Xojo and Real Studio reference materials and has many years of experience in the programming industry. Another great reference book is *I Wish I Knew How To ... Program SQLite*.

ISBN: 978-1-927924-13-6