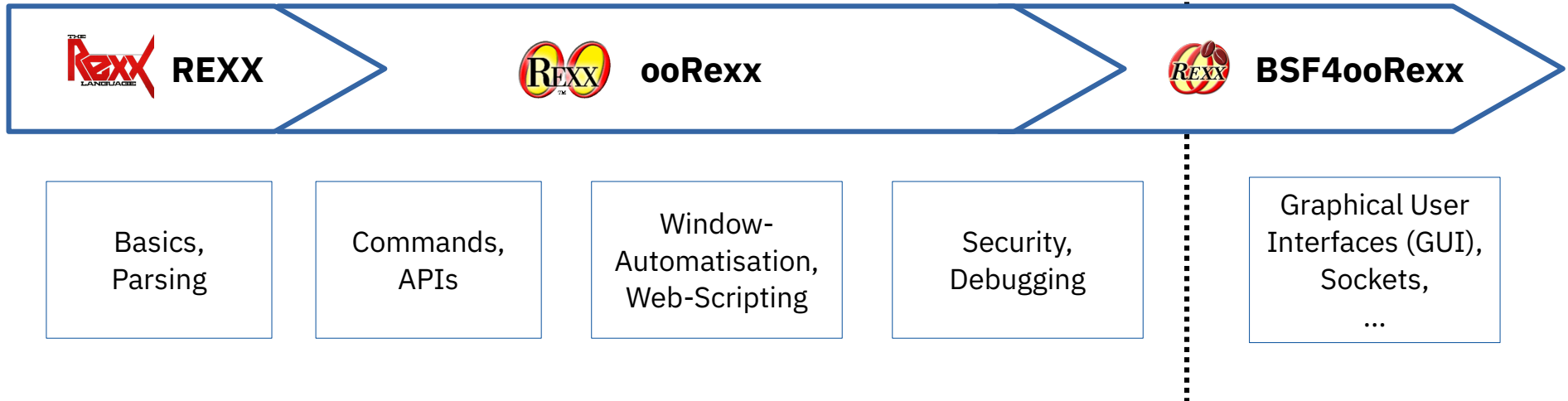


JSON (JavaScript Object Notation)

Business Programming 1

Business Programming 2



- Introduction to **JSON** and the ooRexx package **json.cls**
 - The **json.cls** package got introduced with ooRexx 5.0.0
 - ooRexx 5.1.0 improves it
 - Includes full JSON Boolean type support, handles all ooRexx collection classes
 - Adds the ability to create a legible (human friendly) rendering of JSON text
 - The 5.1.0 version can be used to replace deployed 5.0.0 versions
- Nutshell examples
- Roundup

- JSON got defined in the beginning of the 2000 to ease exchange of structured data via the Internet with JavaScript
- **JSON**
 - Acronym for "JavaScript Object Notation"
 - JSON-Datatypes
 - *Object* – a collection of comma separated name-value pairs (a *Map*) in *curly* brackets
 - *Array* – an ordered list of comma separated values in *square* brackets
 - *String* – a quoted sequence of UTF-8 characters
 - *Boolean* – **true** or **false**
 - *Number* – any number
 - *null* (void) – **null**
 - Cf. <https://www.json.org/>, <https://en.wikipedia.org/wiki/JSON>

The ooRexx package `json.cls`



- Introduced with ooRexx 5.0.0, improved in ooRexx 5.1.0
- Needs to be required, such that its public class `.json` becomes accessible (`::requires "json.cls"`)
 - Explicitly supplies JSON Boolean values with `.json~true` and `.json~false` which behave like ooRexx `.true` and `.false` (can be used as ooRexx Booleans as well)
- Class methods for reading/writing JSON data directly from/to file
 - `fromJsonFile(path2file)`, returns a Rexx object representing the JSON data
 - `toJsonFile path2file, rexxObject [, legible] )`, where `legible` is `.false` (default) or `.true`
- Class (and instance) methods for turning an ooRexx object into a JSON string or a JSON string into an ooRexx object
 - `toJson(rexxObject [, legible])`, where `legible` is `.false` (default) or `.true`
 - `fromJson(jsonString)`, returns a Rexx collection object with the JSON data

Wikipedia JSON Encoded Data (Legible)



Cf. Wikipedia <https://en.wikipedia.org/wiki/JSON> (2025-08-18, "[wikipedia.json](#)")

```
{
  "first_name": "John",
  "last_name": "Smith",
  "is_alive": true,
  "age": 27,
  "address": {
    "street_address": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postal_code": "10021-3100"
  },
  "phone_numbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [
    "Catherine",
    "Thomas",
    "Trevor"
  ],
  "spouse": null
}
```

Nutshell 1: Unencode and Encode JSON Data (1/3)

- Read JSON encoded data from file
 - Use Wikipedia's example
 - JSON encoded sample data is broken up into lines and indented for better legibility
 - Usually JSON encoded data is "minimized", i.e. does not contain ignorable whitespace meant for easing legibility and comprehensibility for humans
 - Result is an ooRexx *directory* object that contains all imported data
 - JSON arrays are represented as ooRexx *array* objects
 - JSON objects (maps) get represented as ooRexx *directory* objects
 - At the end the generated ooRexx *directory* object will be used to create a legible and a minimal JSON rendering of it

Nutshell 1: Unencode and Encode JSON Data (2/3)

```
-- ooRexx 5.1 or higher
parse arg fn          -- get file name
o = .json~fromJsonFile(fn) -- reads and unencodes JSON file

say .json~toJson(o,.true) -- encode as legible (human-friendly) JSON string
say "---"
say .json~toJson(o)       -- encode as minimized (standard) JSON string

::requires "json.cls"    -- get access to the JSON class
```

Nutshell 1: Unencode and Encode JSON Data (3/3)

rexx json_01.rxj wikipedia.json

wikipedia.json

```
{
  "first_name": "John",
  "last_name": "Smith",
  "is_alive": true,
  "age": 27,
  "address": {
    "street_address": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postal_code": "10021-3100"
  },
  "phone_numbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [
    "Catherine",
    "Thomas",
    "Trevor"
  ],
  "spouse": null
}
```

Output (note attributes are sorted by name to ease locating them):

```
{
  "address": {
    "city": "New York",
    "postal_code": "10021-3100",
    "state": "NY",
    "street_address": "21 2nd Street"
  },
  "age": 27,
  "children": [
    "Catherine",
    "Thomas",
    "Trevor"
  ],
  "first_name": "John",
  "is_alive": true,
  "last_name": "Smith",
  "phone_numbers": [
    {
      "number": "212 555-1234",
      "type": "home"
    },
    {
      "number": "646 555-4567",
      "type": "office"
    }
  ],
  "spouse": null
}
```

```
---
{"address":{"city":"New
York","postal_code":"10021-
3100","state":"NY","street_address":"21
2nd Street"},"age":27,"children":
["Catherine","Thomas","Trevor"],"first_nam
e":"John","is_alive":true,"last_name":"Smi
th","phone_numbers":[{"number":"212 555-
1234","type":"home"},{"number":"646 555-
4567","type":"office"}],"spouse":null}
```



Minimal (standard):

- no white space characters
- no carriage-return characters
- no line-feed characters

Example 2: JSON Encoding from ooRexx Data (1/2)

- The `toJSON` class method takes an ooRexx object to encode
- The example creates an ooRexx structure using an ooRexx relation object, an ooRexx array and an ooRexx list and demonstrates how to encode that data
 - Encoding to JSON demonstrates how a JSON Boolean value gets employed
 - The `toJSON` class method creates minimized encodings by default
 - An instance of the JSON class encodes for humans if `toJSON`'s optional `legible` argument is set to `.true`

Example 2: JSON Encoding from ooRexx Data (2/2)

```
rel = .relation~new      -- create a relation (allows duplicates)
rel["WU"]="Vienna Business University"
rel["Wien"] = ("Vienna", "Vienne")  -- English, French
rel["historical districts"] = .list~of(1190, 1090, 1020)
rel["currently in district"] = 1020
rel["is older than Harvard"] = .json~false  -- a JSON false value
```

```
say .json~toJson(rel)      -- encode as minimized JSON (standard)
```

```
say "----"
```

```
say .json~toJson(rel,.true) -- encode as legible JSON (human friendly)
```

```
::requires "json.cls"      -- get access to the JSON class
```

```
{"WU":"Vienna Business University","Wien":
["Vienna","Vienne"],"currently in
district":1020,"historical districts":
[1190,1090,1020],"is older than
Harvard":false}
---
{
  "WU": "Vienna Business University",
  "Wien": [
    "Vienna",
    "Vienne"
  ],
  "currently in district": 1020,
  "historical districts": [
    1190,
    1090,
    1020
  ],
  "is older than Harvard": false
}
```

- ooRexx 5.0 introduced the REXX package `json.cls`
 - Implements a public ooRexx class named `.json`
 - The `fromJSON` class and instance methods allow for turning JSON encoded string data into an ooRexx structure (collection)
 - The `toJSON` class and instance methods allow for encoding any ooRexx structure (collection) into a JSON string
- ooRexx 5.1 improves `"json.cls"`
 - Compatible with ooRexx 5.0 `"json.cls"` (hence no need to change programs)
 - Adds explicit support for JSON Boolean values
 - The attribute `legible`, if set to `.true` will encode JSON in a human friendly form
 - Adds utility class methods to directly read (`fromJsonFile(fileName)`) from or write to files (`toJsonFile(fileName, rexxObject[, isLegible=.false])`)