

Hands on JSON



Octavian Nadolu, Syncro Soft
octavian_nadolu@oxygenxml.com
@OctavianNadolu

schematron
Structured
editing
XML
review
XQuery
Publish
PDF
DTD
DocBook
oxygen
authoring
XML Editor
XSD SCHXSD Single
XPRRNCFO
frameworks
Profiling
WSDL
styles
visual
WebHelp
DITA
TEI
XSL
PHP
Ant
Js
JS
KML
XSLT
SVN
JSON
SVG
IDREFS
WebDAV
DocBook
XML Editor
Single
Source
Database
XHTML
Cha
Col
Wa

© 2019 Syncro Soft SRL. All rights reserved.

About me

Software Architect at Syncro Soft

octavian.nadolu@oxygenxml.com

- 15+ years of XML technology experience
- Contributor for various XML-related open source projects
- Editor of Schematron QuickFix specification developed by a W3C community group



schematron
Structured
editing
XML
review
XQuery
Publish
PDF
IDREFS
WebDAV
DTD DocBook
oxygen
authoring
XML Editor
XSD SCHXSD Single
XPRRNC FO
frameworks
Profiling
WSDL
styles
visual
WebHelp
DITA
TEI
XSL
PHP
Ant
Js

JS
KML
XSLT
SVN
JSON
SVG
WebDAV
DocBook
authoring
XML Editor
Single
Source
Datab
XHTML
Cha
Col
We

Agenda

- JSON Documents Structure
- Validating JSON Documents
- Querying and Transforming JSON Documents
- Converting JSON Documents



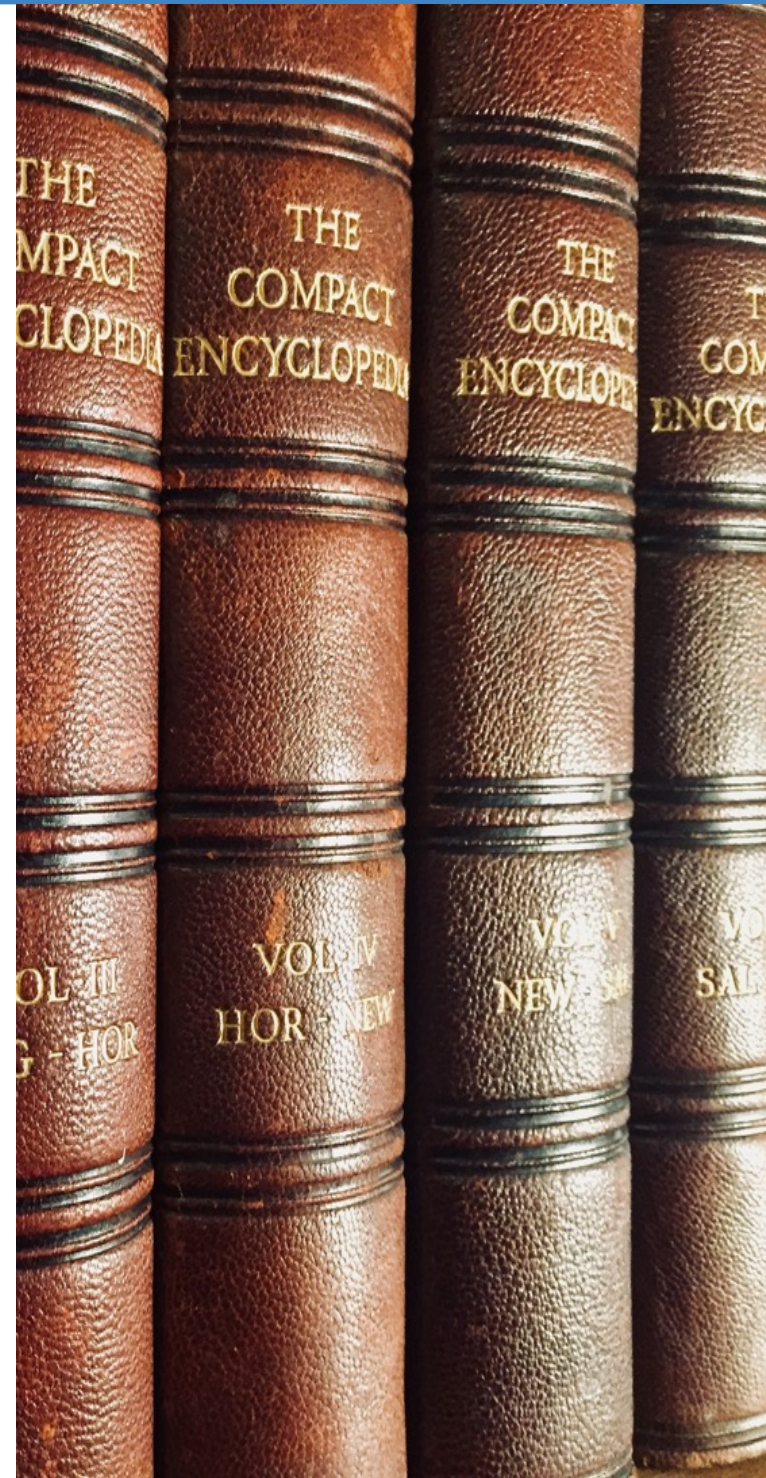
JSON

(JavaScript Object Notation)

- Data representation format
- Used for API and Configs
- Lightweight and easy to read/write

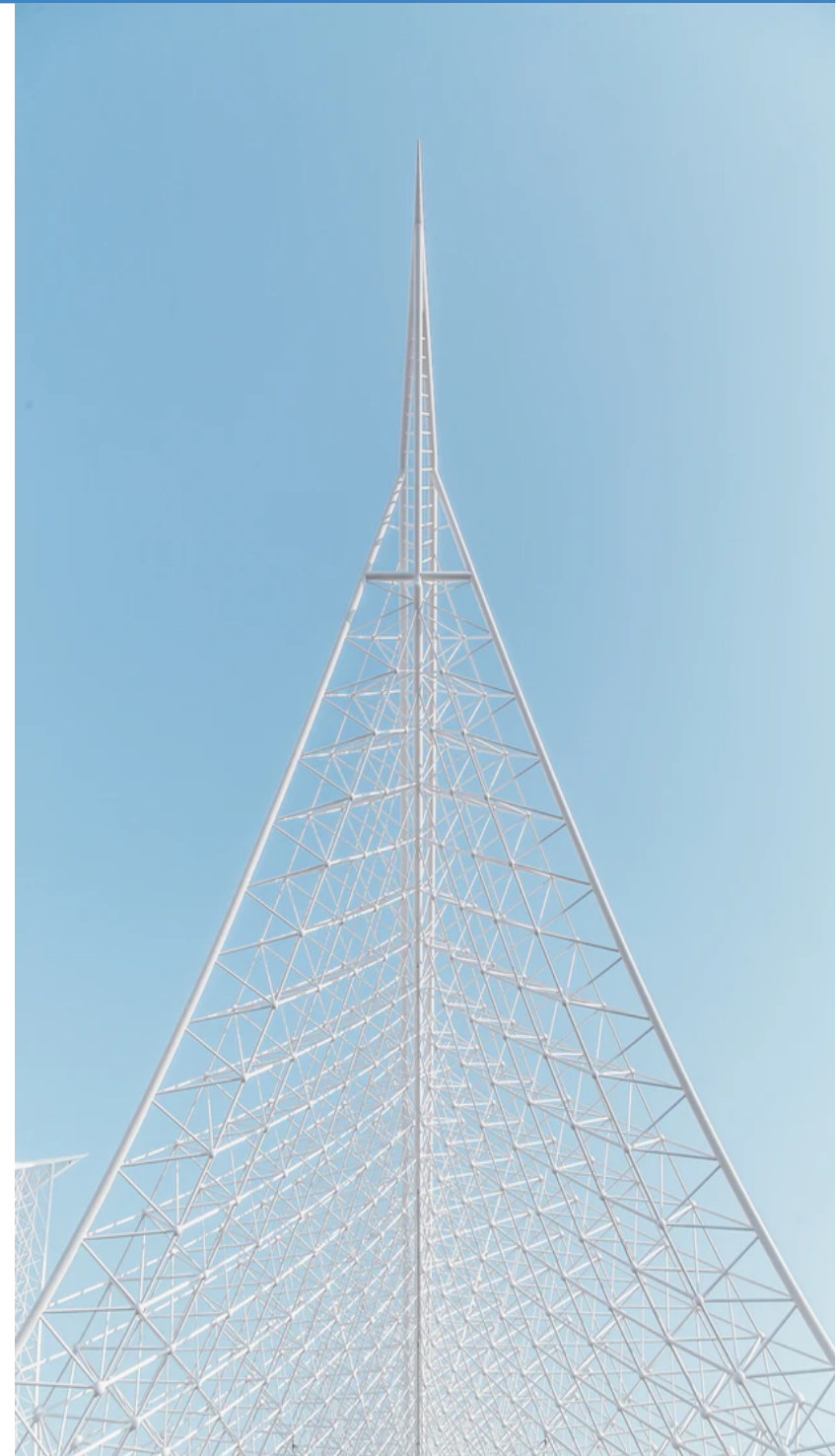


<http://json.org>



JSON Documents Structure

- JSON Data – name/value pair
- JSON Object {...}
- JSON Array [...]



JSON Data

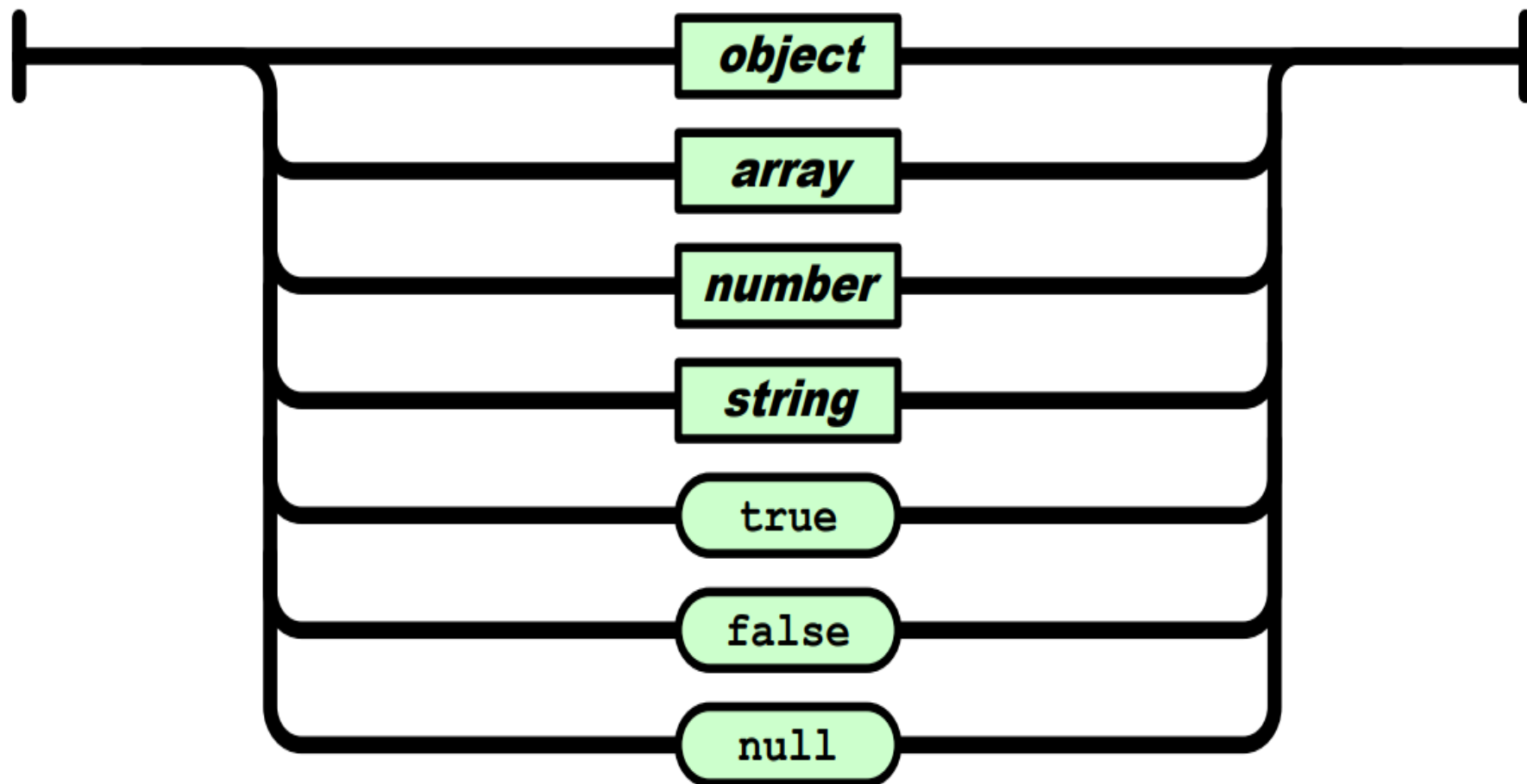
```
"firstname": "Jane"
```

- A collection of name/value pairs
- The name in double quotes, followed by a colon, followed by a value

Value

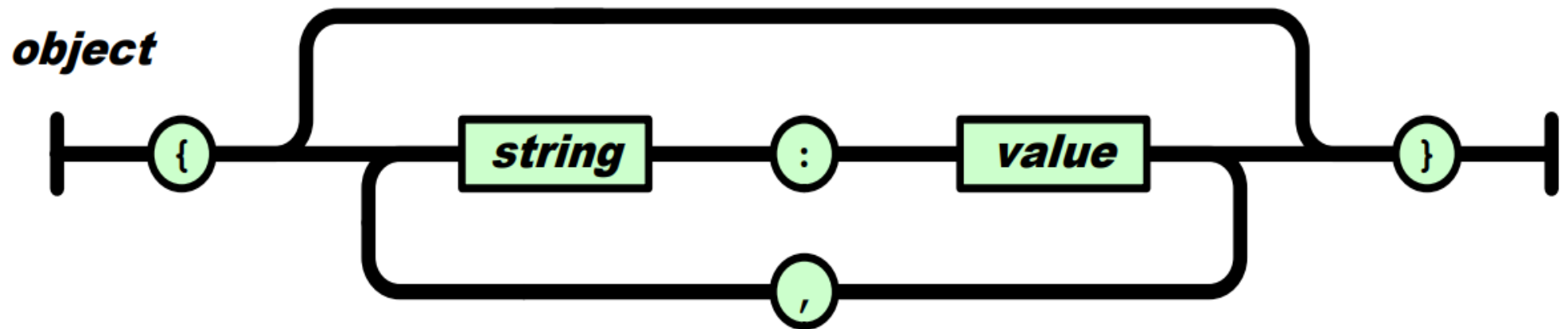
`{...}`, `[...]`, `2`, `"one"`, `true`, `null`

value



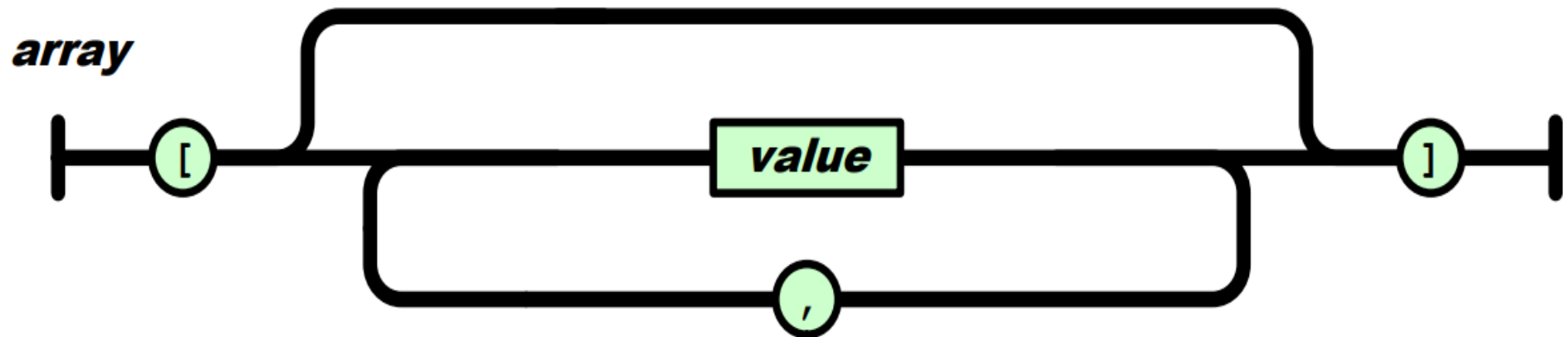
Object

```
{"firstname": "Jane", "lastname": "Doe"}
```



Array

```
[{"firstname": "Jane", "lastname": "Doe"},  
 {"firstname": "John", "lastname": "Jones"}]
```



Example

```
{  
  "persons": [  
    {  
      "id": "jane.doe",  
      "firstname": "Jane",  
      "lastname": "Doe",  
      "email": "jane@oxygenxml.com",  
      "age": 37  
    },  
    {  
      "id": "john.jones",  
      "firname": "John",  
      "lastname": "Jones",  
      "email": "john@oxygenxml.com",  
      "age": 42  
    }  
  ]  
}
```

The diagram illustrates the JSON structure with arrows pointing to labels:

- A black arrow points from the opening curly brace `{` to the label **Object**.
- A green arrow points from the opening square bracket `[` to the label **Array**.
- A blue arrow points from the comma `,` after the first object to the label **Property**.
- An orange arrow points from the value `37` to the label **Value**.

Validating JSON Documents

- Checking **Well-Formedness** in JSON Documents
- **Validating JSON** Documents Against JSON Schema
- **Validating JSON Schema** According to the Specification

Checking Well-Formedness

Check if the JSON document respects the JSON specification

```
{  
  "id": "jane.doe",  
  "firstname": "Jane",  
  "lastname": "Doe",  
  "email": "jane@oxygenxml.com",  
  "age": 37  
}
```

ECMA-404 The JSON Data Interchange Standard

<http://json.org/>

Example

```
{  
  "persons" [  
    {  
      "id": "jane.doe",  
      "firstname": "Jane",  
      "lastname": "Doe",  
      "email": "jane@oxygenxml.com",  
      "age": 37,  
    },  
    {  
      "id": "john.jones",  
      "firname": "John",  
      "lastname": "Jones",  
      "email": "john@oxygenxml.com",  
      "age": 42  
    }  
  ]  
}
```

Expected a ':' after a key

Expected ',' character

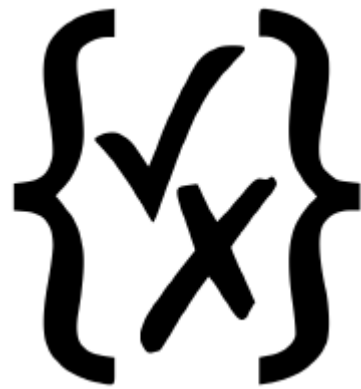
Unexpected ',' character

Expected quotes

Expected '}'

JSON Schema

JSON Schema is a vocabulary that allows you to **annotate** and **validate** JSON documents



<http://json-schema.org>



Defining JSON Schema

- Similar to XML Schema, RNG, or DTD
- Written in JSON
- Used to define the structure of a JSON data

```
{"type": "string"}
```

JSON Schema

```
"I'm a string"
```

JSON Instance

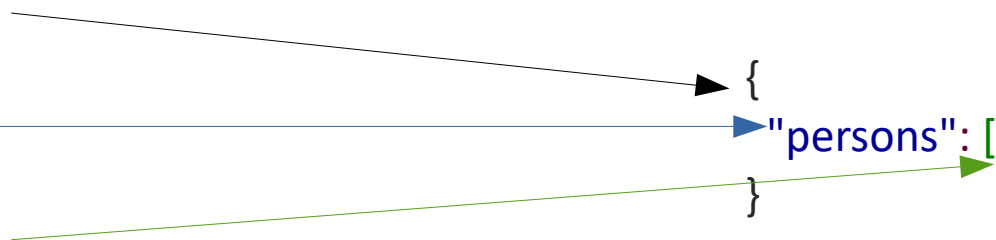
Example

JSON Schema

```
{  
  "type": "object",  
  "properties": {  
    "persons": {  
      "type": "array"  
    }  
  }  
}
```

JSON Instance

```
{  
  "persons": [ ]  
}
```



Example

JSON Schema

```
{
  "type": "object",
  "properties": {
    "persons": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": {"type": "string"},
          "firstname": {"type": "string"},
          "lastname": {"type": "string"},
          "email": {"type": "string", "format": "email"},
          "age": {"type": "number"}
        }
      }
    }
  }
}
```

JSON Instance

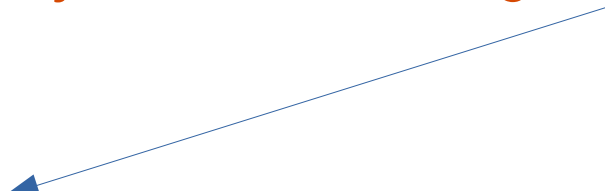
```
{
  "persons": [
    {
      "id": "jane.doe",
      "firstname": "Jane",
      "lastname": "Doe",
      "email": "jane@oxygenxml.com",
      "age": 37
    }
  ]
}
```



JSON Schema Definition

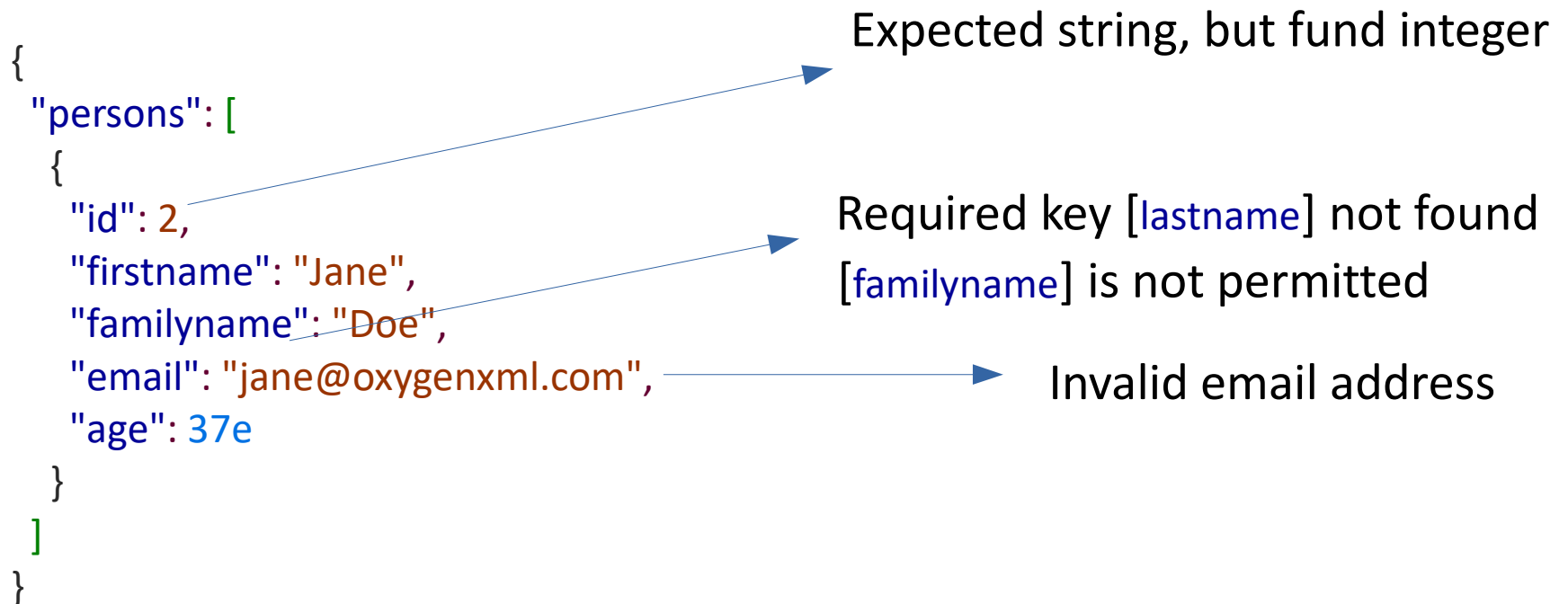
- It is recommended to have the schema definition on the first level

"\$schema": "<http://json-schema.org/draft-07/schema#>"

- JSON Schema used versions:
 - Draft 4
 - Draft 6
 - Draft 7
 - Draft 8
- 

Validating with JSON Schema

- Check if the JSON instance respects the definitions

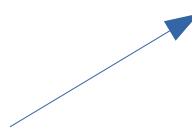


Associate JSON Schema

- Associating a Schema to JSON Documents
 - Directly in JSON document – using `$schema` property
 - In application options

Absolute or relative URI

```
{  
  "$schema": "person-schema.json",  
  "persons" [ ... ]  
}
```



Validating JSON Schema

- Check for well-formedness
- Validate accordingly to the Internet Engineering Task Force (IETF) Specification



Validating JSON with Schematron

- Specify the JSON structure using JSON Schema
- Express co-constraints and define custom rules using Schematron



Schematron for JSON

- Use XPath to express the rules

```
<sch:rule context="persons">  
  <sch:assert test="number(age) > preceding-sibling::node()/age ">  
    The person age must be grater than the previous person age </sch:assert>  
</sch:rule>
```

Schematron using JSONPath

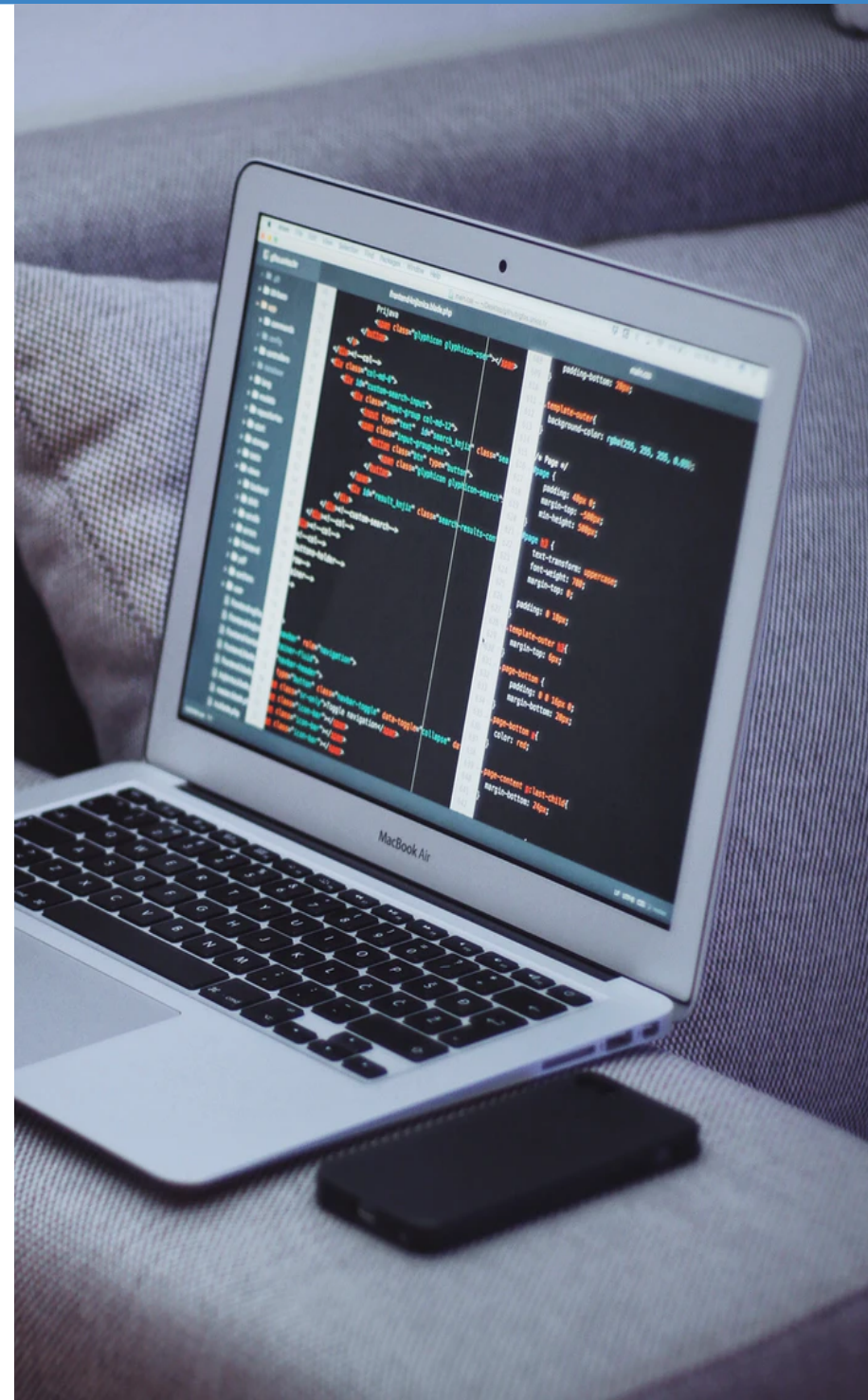
- Schematron reimaged for JSON with no whiff of XML/XPath
- Moved out of the XPath and use JSONPath

```
"rule": [  
  {"context": "$.persons.*",  
    "assert": [  
      {"test": "(jp.query(contextNode,'$..age') >= 21",  
        "message": "The person age must be grater than 21"  
      }  
    ]  
  }  
]
```

<https://github.com/amer-ali/jsontron>

Querying and Transforming

- Query using JSON Pointer, JSONiq, or XPath
- Transform JSON using JavaScript, XSLT, XQuery



JSON Pointer

Used to identify a specific value within a JSON document

`#/persons/0/email`

Matches first person email

Specification

<https://tools.ietf.org/html/rfc6901>

JSONiq

A query and processing language for JSON

www.jsoniq.org

```
let $stats := collection("stats")
for $access in $stats
group by $url := $access.url
return
{
  "url": $url,
  "avg": avg($access.response_time),
  "hits": count($access)
}
```

XPath

- Powerful language for queering XML documents
- Was adopted by applications also for JSON

`/persons[2]/email`

Matches second person email

Process JSON with JavaScript

- Most popular way to process JSON documents
- Multiple libraries

```
var json = '{"name":"John Doe", "age":42}';  
obj = JSON.parse(json);
```

```
console.log(obj.name); // Result John Doe  
console.log(obj.age); // Result 42
```

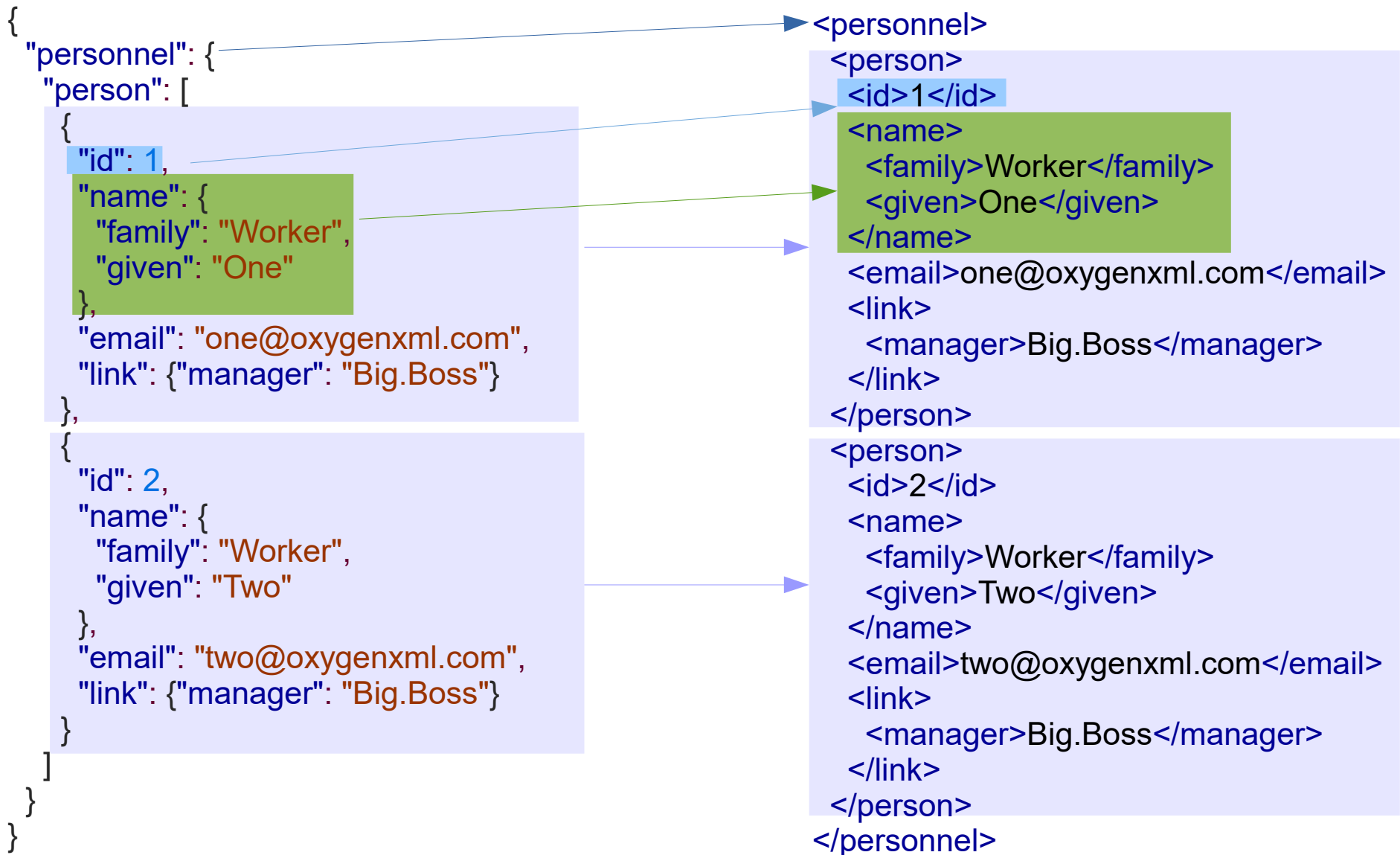
Transform using XSLT/XQuery

- Transform JSON Documents to Different Formats
- Different functions to process JSON document:
 - `json-doc($href as xs:string?) as item()?`
 - `json-to-xml($json-text as xs:string?) as document-node()?`

Converting JSON Documents

- JSON to XML and XML to JSON
- Convert using XSLT
- Online Converters

JSON to XML



JSON to XML Conversion Details

- A `<JSON>` element is added as root if the converted JSON has multiple properties on the first level

```
{  
  "person": "one",  
  "id": "personnel-id"  
}
```

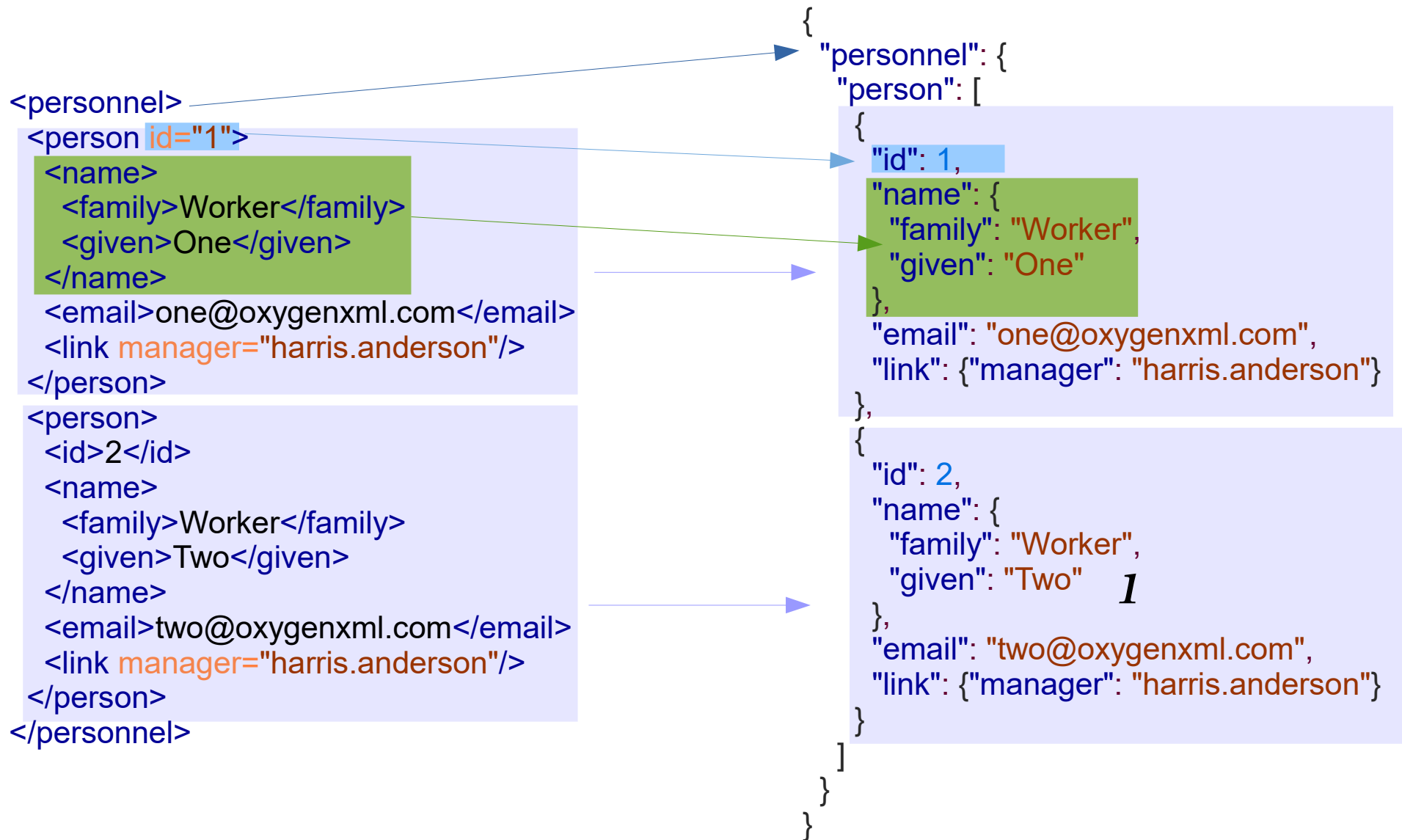
```
<JSON>  
  <person>one</person>  
  <id>personnel-id</id>  
</JSON>
```

- An `<array>` element is added as root if the converted JSON is an array

```
[  
  {"name": "Boss"},  
  {"name": "Worker"}  
]
```

```
<array>  
  <array>  
    <name>Boss</name>  
  </array>  
  <array>  
    <name>Worker</name>  
  </array>  
</array>
```

XML to JSON



XML to JSON Conversion Details

- XML attributes are converted to properties

```
<person id="1"/>
```

```
{"person": {"id": 1}}
```

- Multiple elements with the same name are converted into an array

```
<person/> <person/>
```

```
"person": [ "", "" ]
```

- Text in mixed content will be converted in a **#text** property

```
<p>This is an <b>example</b>!</p>
```

```
p: {"#text": "This is an", "b": "example", "#text1": "!"}
```

Convert using XSLT

- Library that converts XML to JSON using XSLT

<https://github.com/bramstein/xsltjson>

XSLT Specification for JSON

<http://www.w3.org/TR/xslt-30/#json>

Online Converter

- Online XML to JSON Converters

The screenshot shows an online converter interface with two main panels: XML and JSON. The XML panel on the left is titled 'XML' and has a 'Choose File' button next to 'sample.xml'. It contains a code editor with the following XML code:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <article xmlns="http://docbook.org/ns/d
3   xmlns:xlink="http://www.w3.org/1999
4   <info>
5     <title>Welcome to DocBook Suppo
6   </info>
7   <sect1>
8     <title>Inline Markup and Images
9     <para>This sample shows that &l
10      with the
11      dockbookx.dtd<?oxy_delete a
12     <para>The following <code>Docbo
13     transformation scenario. Fo
```

The JSON panel on the right is titled 'JSON' and has a 'Choose File' button next to 'No file chosen'. It contains a code editor with the following JSON code:

```
1 {
2   "article": {
3     "xmlns": "http://docbook.org/ns/docb
4     "version": "5.0",
5     "xmlns:xlink": "http://www.w3.org/19
6     "info": {"title": "Welcome to DocBoo
7     "sect1": [
8       {
9         "title": "Inline Markup and Imag
10        "para": [
11          [
12            "This sample shows that <oxy
13          ]
14        ]
15      }
16    ]
17  }
18 }
```

Between the two panels are two circular arrows: a right-pointing arrow above and a left-pointing arrow below. At the bottom of each panel are three buttons: 'Download', 'Copy', and 'Clear'.

Convert between XML and JSON

Conclusion

- JSON conforms to ECMA/ISO specification
- Validate JSON with JSON Schema
- Query and process JSON using JSON Pointer, JSONiq, XPath, or JavaScript and XSLT
- Conversions to convert between JSON and XML





Questions?

Octavian Nadolu
Software Architect at Syncro Soft

octavian.nadolu@oxygenxml.com
Twitter: @OctavianNadolu
LinkedIn: octaviannadolu

<https://sundt01.honestly.de>

