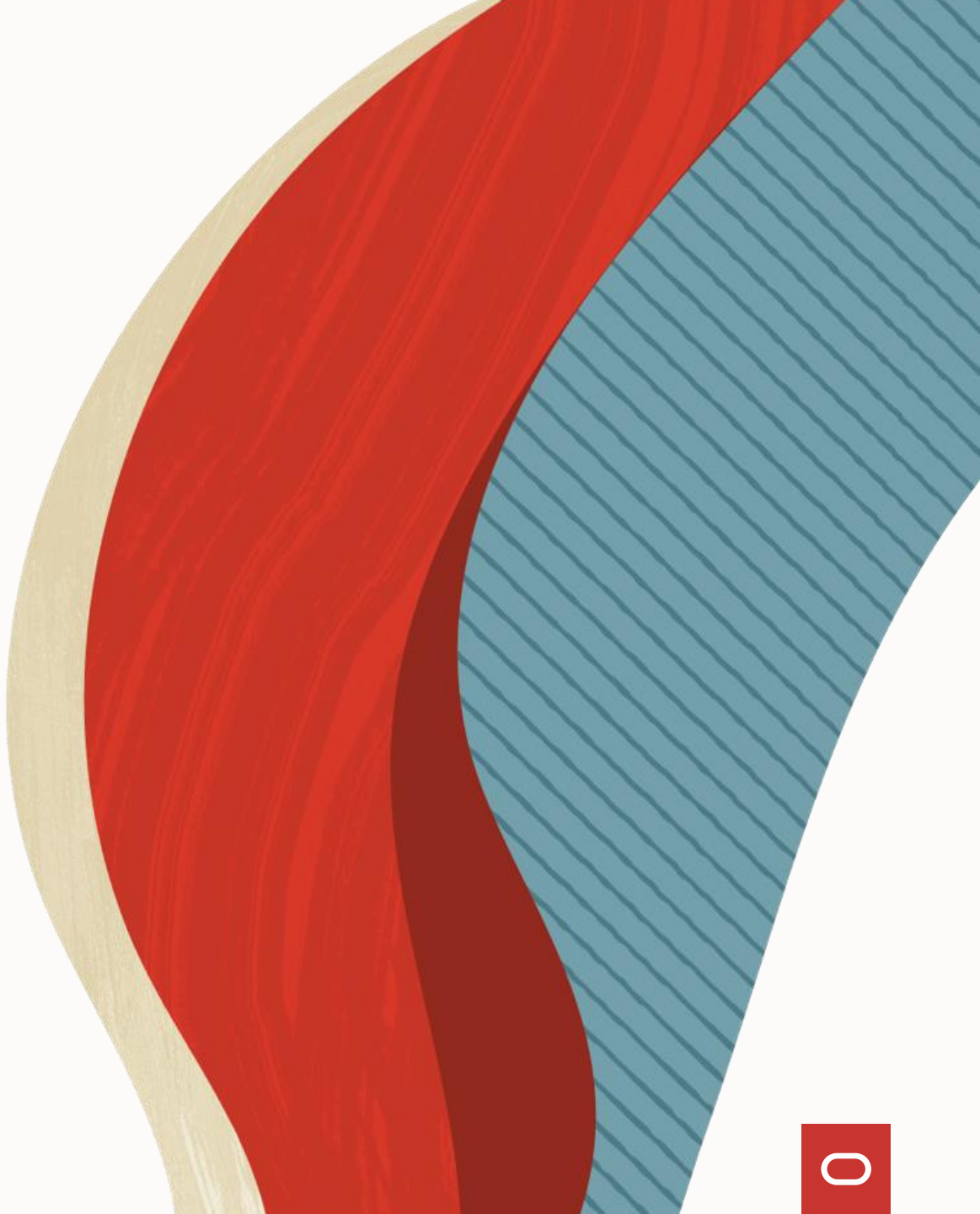
Oracle Database 23ai

XML and JSON Document Convergence

Witold Swierzy
Oracle EMEA JSON Database Expert
witold.swierzy@oracle.com

August 2025



Document formats supported by Oracle Database

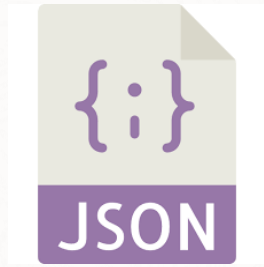
- XML
  - Stands for Extensible Markup Language
  - Defined in 1996 by W3C consortium
  - Self-descriptive and extensible format
  - XSLT transformation part of XML standard
  - Xquery, Xpath languages used to query XML documents
  - Supported by Oracle Database in XML DB feature
  - Basic type used to store and process XML data: XMLType
  - Rich PL/SQL API

Document formats supported by Oracle Database

- JSON
  - Stands for JavaScript Object Notation
  - Open standard and data interchange format
  - Data are stored as key:value pairs or in form of arrays
  - Originally derived from JavaScript
  - Supported by Oracle Database as a complete data model
    - Specialized datatypes and data structures
      - LOB/JSON/JSON_ELEMENT_T
      - JSON Collection tables/views/duality views
  - Rich PL/SQL API
  - MongoDB API available to use Oracle Database as MongoDB JSON specialized database engine

Document format duality

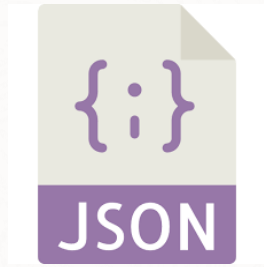
Challenges



- Different applications use different data formats
  - Data sharing issues
  - Lack of single point of truth
  - High costs of maintenance
- Application modernization
  - JSON is usually considered as more modern standard
  - JSON is lightweight
- Data format not compatible with implemented ETL/ELT processes
  - Transformations and schemas implemented in XML (XSLT, XML Schemas)
  - Incoming data in JSON format

Document format duality

Solution



Document Format Converter

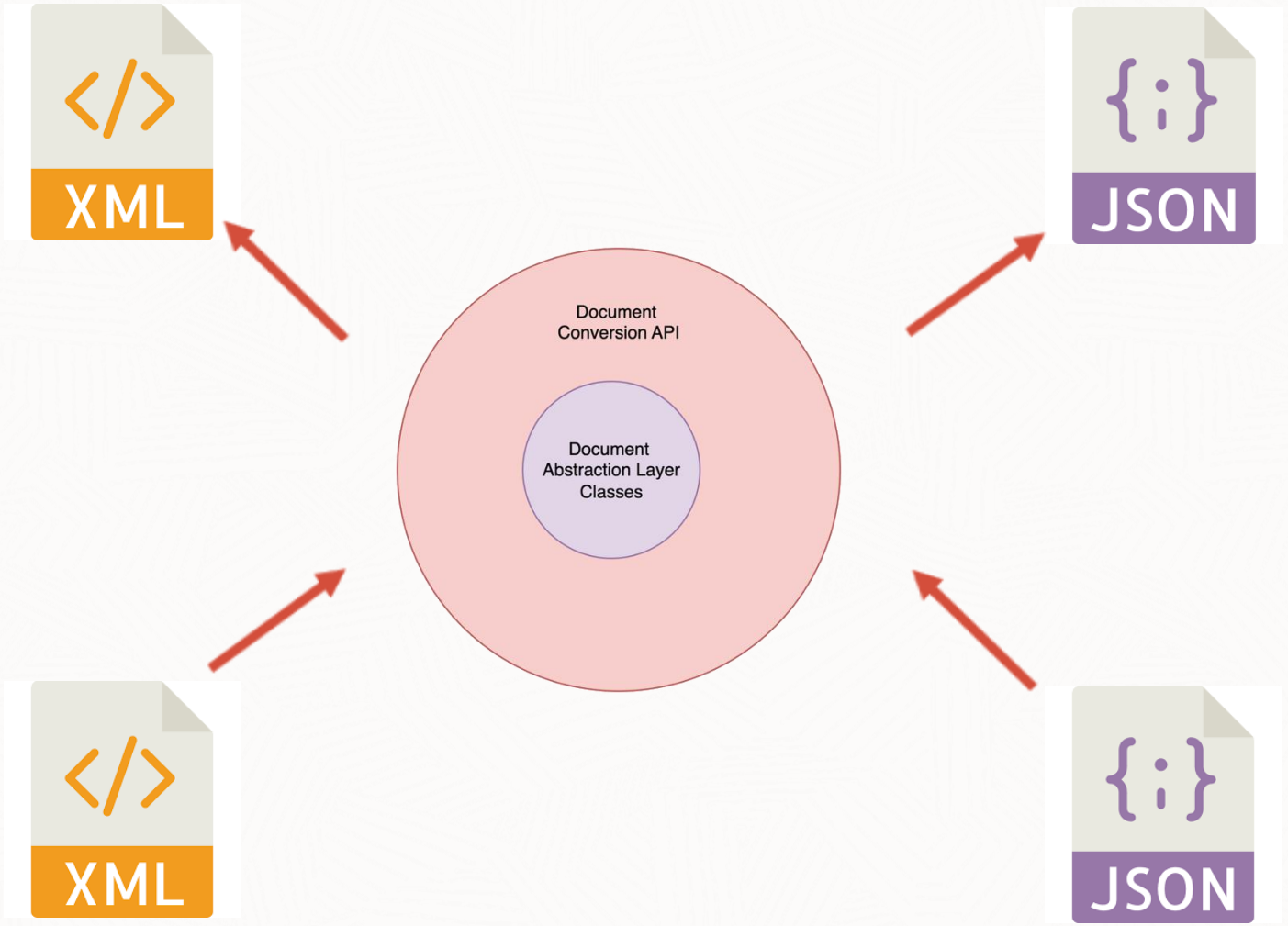
- PL/SQL API allowing for on-the-fly transformation between formats
- Implemented in form of object types and packages
- Stored in the database – as near to the data as possible

Plays the role of abstraction layer

- Makes a document-oriented application totally independent of a data format
- Different applications can see the same data in different formats

Document Format Converter

Architecture



Document Format Converter

Document Abstraction Layer

- Low-level interface used to build a layer, which does not depend on specific format
- Makes a document-oriented application totally independent of a data format
- Different applications can see the same data in different formats
- Main components
  - **DocElement**
    - Main class of the interface
    - Constructors allowing to build object based on XMLType and JSON_ELEMENT_T data
    - getAsXML and getAsJSON methods
  - **DocAttribute**
    - Allows for storing and processing XML attributes
  - **DOC_CONV_UTL PL/SQL package**
    - Low-level procedures and functions used internally by DocElement and DocAttribute classes

Document Format Converter

Document Abstraction Layer

Example #1: XML to JSON conversion

```
SQL> set serveroutput on
SQL>
SQL> declare
  2     xDoc XMLType         := XMLType('<main>
  3                                        <a>val_a</a>
  4                                        <b>val_b</b>
  5                                      </main>');
  6     eDoc DocElement      := DocElement(xDoc);
  7     jDoc JSON_ELEMENT_T := eDoc.getAsJSON;
  8  begin
  9     dbms_output.put_line(jDoc.to_String);
 10  end;
 11* /
{"main":{"a":"val_a","b":"val_b"}}
```

Document Format Converter

Document Abstraction Layer

Example #2: JSON to XML conversion

```
SQL> set serveroutput on
SQL>
SQL> declare
  2     jDoc JSON_ELEMENT_T := JSON_ELEMENT_T.parse('{"a":{"b":"c"}}');
  3     eDoc DocElement     := DocElement(jDoc);
  4     xDoc XMLType        := eDoc.getAsXML;
  5  begin
  6     dbms_output.put_line(xDoc.getClobVal);
  7  end;
  8* /
<a><b>c</b></a>
```

Document Format Converter

Document Conversion API

- High-level API allowing for easy conversion between supported formats
- Basic assumptions
  - Ease of use
  - Simplicity
  - Flexibility
- Single PL/SQL package
- doc_conv

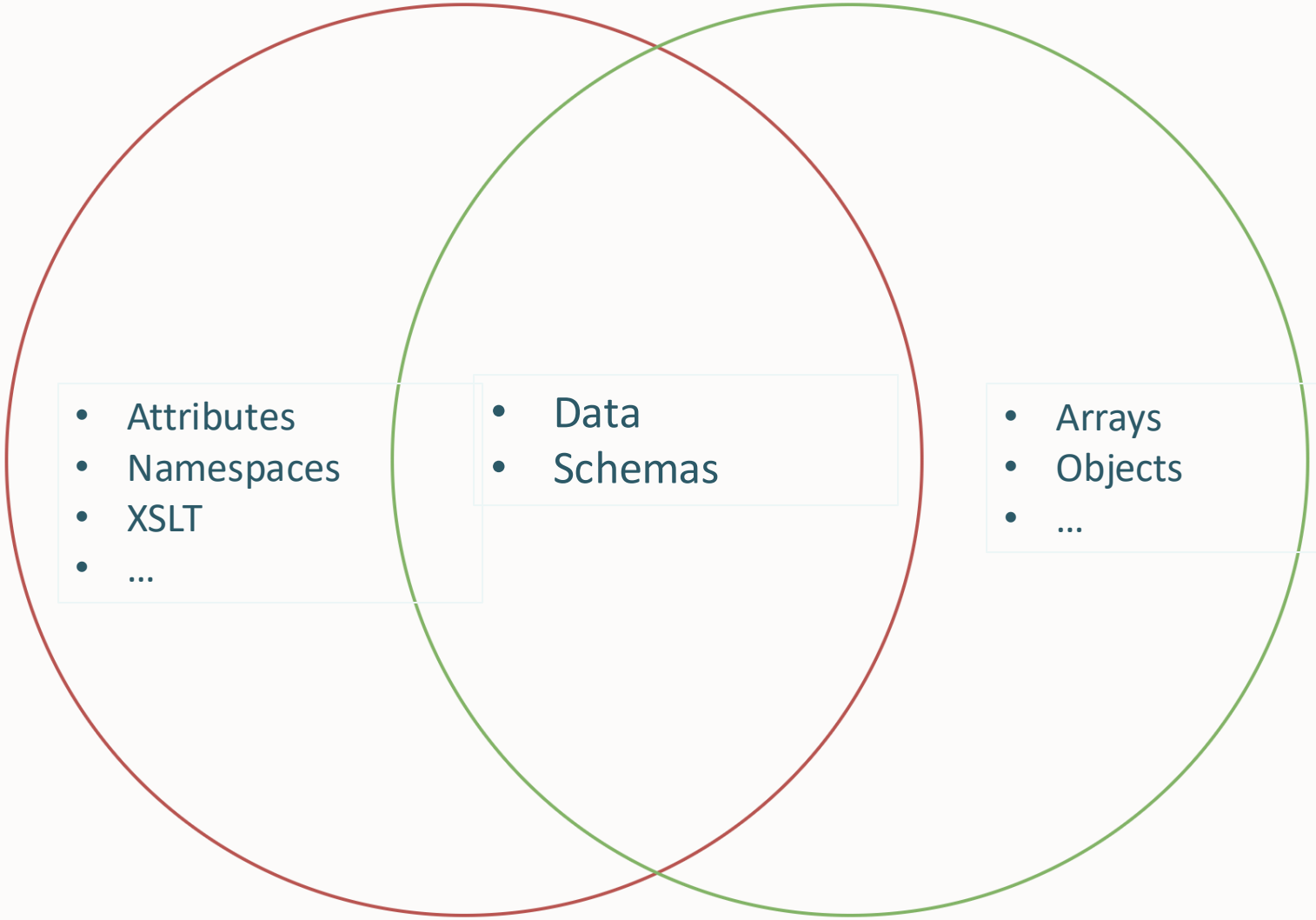
Document Format Converter

Document Conversion API

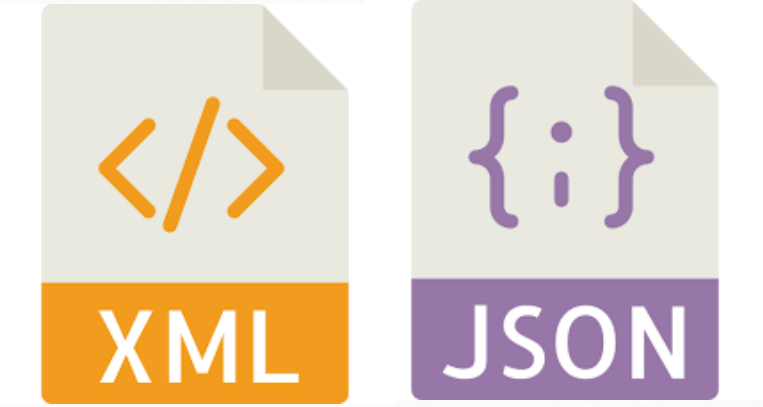
DOC_CONV PL/SQL package

```
SQL> desc doc_conv

FUNCTION JSON2XML  RETURNS OPAQUE/XMLTYPE
  Argument Name          Type              In/Out Default?
  ---------------------- ----------------- ------ --------
  JDOC                   JSON              IN

FUNCTION JSON_ELEMENT_T2XML  RETURNS OPAQUE/XMLTYPE
  Argument Name          Type              In/Out Default?
  ---------------------- ----------------- ------ --------
  JDOC                   OBJECT            IN

FUNCTION XML2JSON  RETURNS JSON
  Argument Name          Type              In/Out Default?
  ---------------------- ----------------- ------ --------
  XDOC                   OPAQUE/XMLTYPE    IN

FUNCTION XML2JSON_ELEMENT_T  RETURNS OBJECT
  Argument Name          Type              In/Out Default?
  ---------------------- ----------------- ------ --------
  XDOC                   OPAQUE/XMLTYPE    IN
```

XML vs JSON
Challenges



XML vs JSON

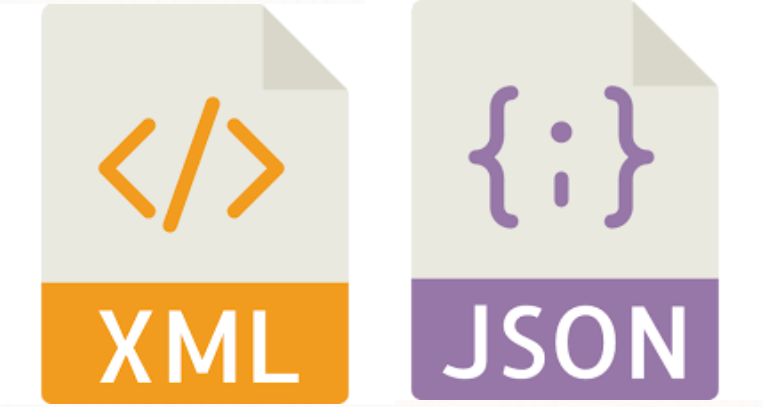


Solution: Supplemental Elements

- Elements with parametrized values
- Added automatically when needed to ensure correct syntax
- Customizable
- Example : JSON objects

```
SQL> set serveroutput on

SQL> declare
  2     jd JSON_ELEMENT_T := JSON_ELEMENT_T.parse('{"key01":"val01","key02":"val02"}');
  3     ed DocElement := DocElement(jd);
  4     xd XMLType := ed.getAsXML;
  5  begin
  6     dbms_output.put_line(xd.getClobVal);
  7  end;
  8* /
<_dc_object_><key01>val01</key01><key02>val02</key02></_dc_object_>

PL/SQL procedure successfully completed
```

XML vs JSON



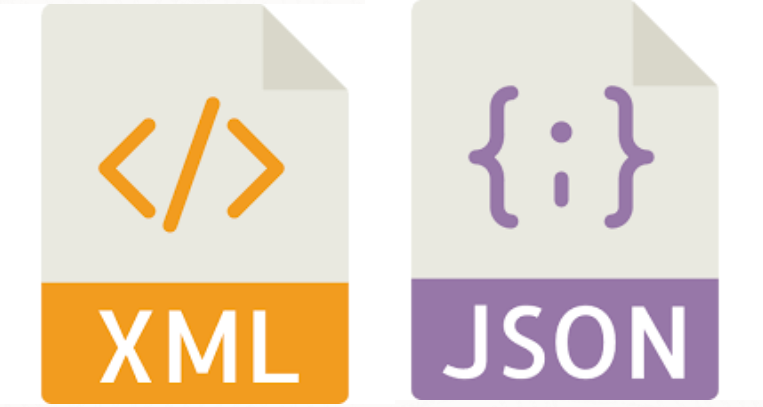
Supplemental Elements:

- JSON Objects
- JSON Arrays
- XML Comments and CDATA elements
- XML Attributes

Parameter name	Description	Default Value
XML_ARRAY_NAME	Element storing arrays of values in XML document – equivalent of JSON arrays	_dc_array_
XML_LIST_NAME	Element storing objects in XML document – equivalent of unnested JSON element	_dc_object_
JSON_ATTR_NODE	Element allowing for storing XML attributes in JSON documents	_dc_attribute_
JSON_COMMENT	Element allowing for storing XML comments in JSON documents	_dc_comments_
JSON_CDATA	Element allowing for storing XML CDATA elements in JSON documents	_dc_cdata_

XML vs JSON

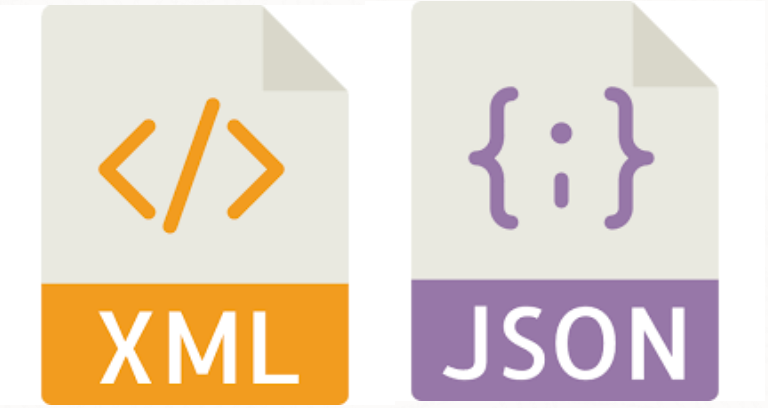
Supplemental Elements, example 2




```
SQL> declare
  2    jd JSON_ELEMENT_T := JSON_ELEMENT_T.parse('[1,2]');
  3    ed DocElement := DocElement(jd);
  4    xd XMLType := ed.getAsXML;
  5  begin
  6    dbms_output.put_line(jd.to_String);
  7    dbms_output.put_line(xd.getClobVal);
  9  end;
 10* /
<_dc_array_><_dc_item_>1</_dc_item_><_dc_item_>2</_dc_item_></_dc_array_>

PL/SQL procedure successfully completed.
```

XML vs JSON

Supplemental Elements, example 2
modification of default settings




```
SQL> declare
  2    jd JSON_ELEMENT_T := JSON_ELEMENT_T.parse('[1,2]');
  3    ed DocElement;
  4    xd XMLType;
  5  begin
  6    doc_conv.set_param('XML_ARRAY_NAME','int_array');
  7    doc_conv.set_param('XML_ITEM_NAME','int_val');
  8    ed := DocElement(jd);
  9    xd := ed.getAsXML;
 10    dbms_output.put_line(xd.getClobVal);
 11  end;
 12* /

<int_array><int_val>1</int_val><int_val>2</int_val></int_array>
```

Document format duality

To be implemented in future releases

- XML Namespaces and XML Transformation support
  - Planned to be implemented in the next release
- Support for other document formats