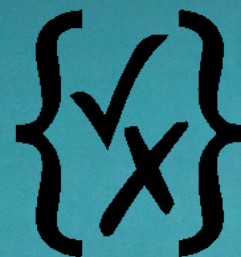


Create and Design JSON Schema

Octavian Nadolu, Syncro Soft
octavian.nadolu@oxygenxml.com
@OctavianNadolu



© 2022 Syncro Soft SRL. All rights reserved.

Agenda

- JSON Schema Specification
- Changes in JSON Schema version 2020-12
- Design JSON Schema From Scratch
- Edit and Refactor Complex JSON Schemas
- Generate Documentation
- JSON Schema in OpenAPI/AsyncAPI

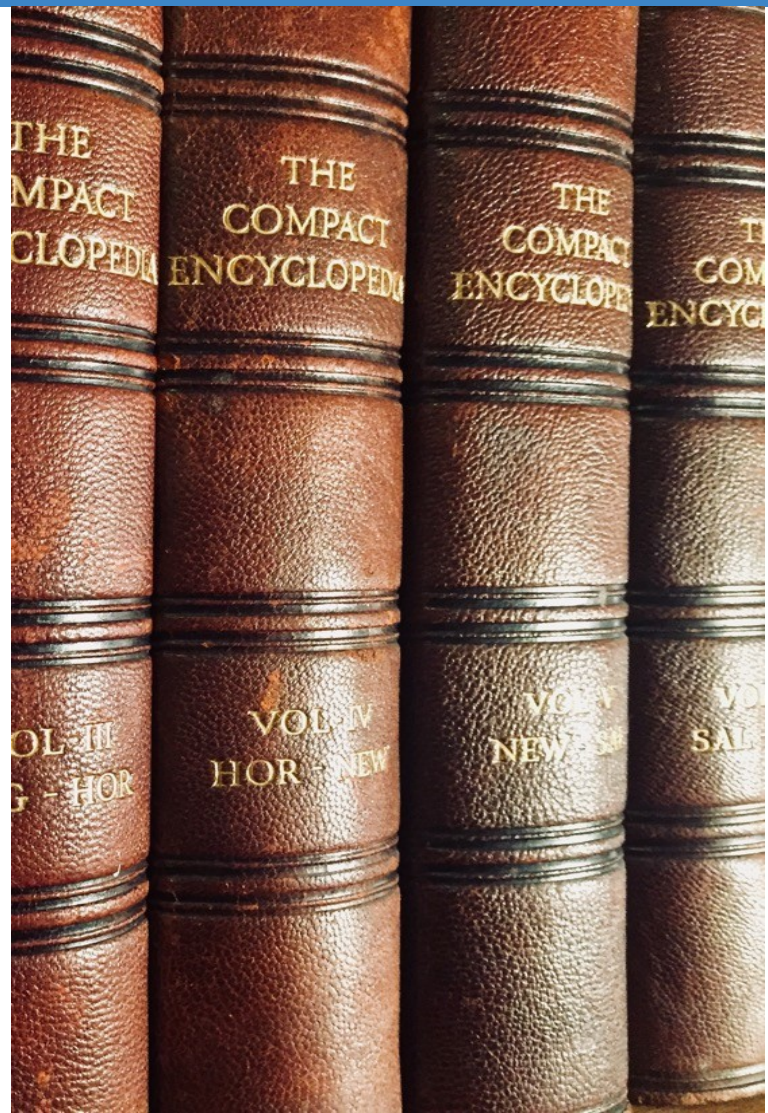


JSON Schema

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



<http://json-schema.org>



JSON Schema Benefits

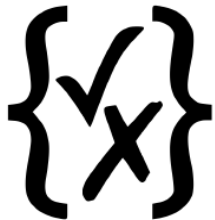
- Describe your data format or API
- Provide human and machine readable documentation
- Validate data
- Automated testing



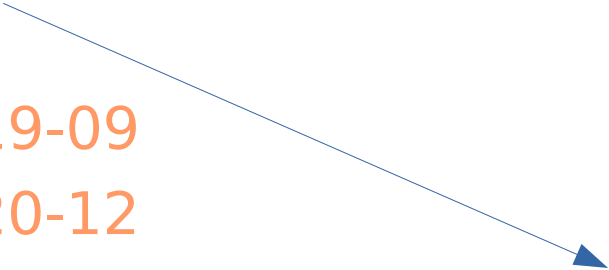
JSON Schema

JSON Schema Usage

- Describe the structure and validation constraints of JSON documents
- JSON Schema is used in OpenAPI specification
- JSON Schema partially used in AsyncAPI specification
- JSON Schema is used in databases



JSON Schema Definition

- JSON Schema used versions:
 - draft-04
 - draft-06
 - draft-07
 - version 2019-09
 - version 2020-12
- 

`"$schema": "http://json-schema.org/draft-07/schema#"`

It is recommended to have the schema definition on the first level in the JSON document

JSON Schema Sample

- Basic JSON Schema sample

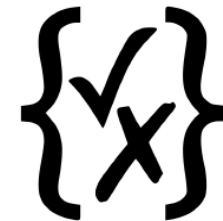
```
{  
  "$schema": "http://json-schema.org/draft/2020-12/schema#",  
  "type": "object",  
  "properties": {  
    "productId": {  
      "type": "integer"  
    }  
  }  
}
```

JSON Instance:

```
{  
  "productId": 1  
}
```

JSON Schema 2019-09/2020-12 Changes

- `definitions` → `$defs` (renamed)
- `items` → `prefixItems` (renamed and updated)
- `additionalItems` → `items` (renamed)
- `dependencies` → `dependentSchemas` + `dependentRequired` (split)
- `unevaluatedProperties` and `unevaluatedItems` (new)
- `maxContains` and `minContains` (new)
- `$anchor` (new)
- `$dynamicAnchor` and `$dynamicRef` (new)
- `$vocabulary` (new)



<https://json-schema.org/specification.html>

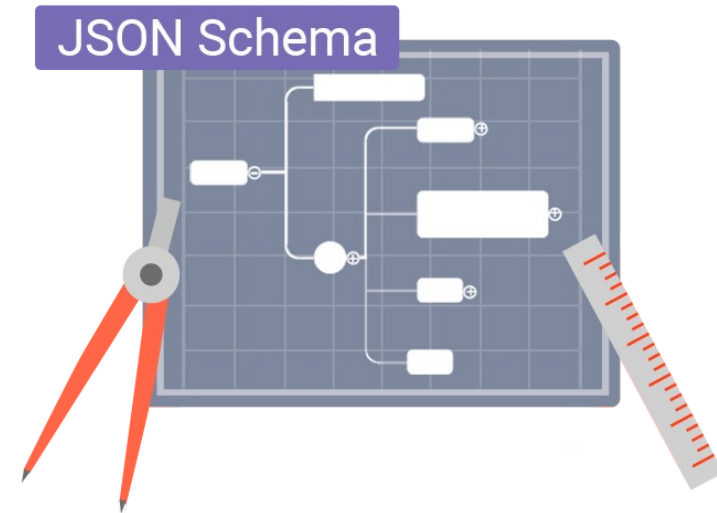
Question: Do you use JSON Schema, what versions?

- No
- Yes. Draft 4, 6 or 7
- Yes. Version 2019-09, 2020-12
- Yes. Other (use the Questions pane to provide more details)



JSON Schema Support in Oxygen

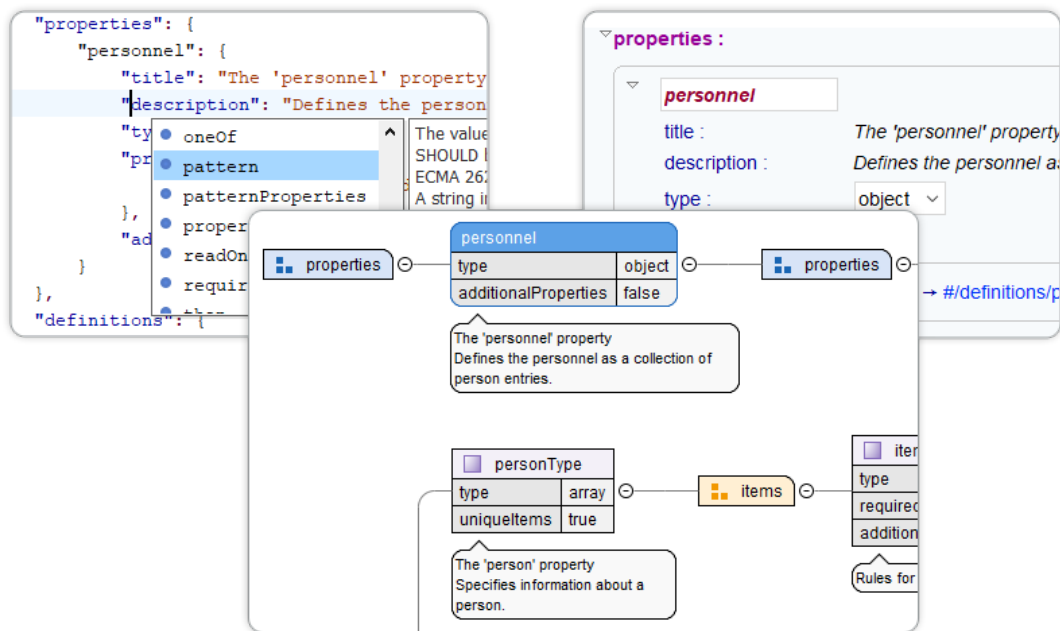
- **JSON Schema Editor** - specialized editor with various editing features
- **Validation** against JSON Schema
- **Editing** based on JSON Schema
- **Tools**
 - Generate **JSON Schema Documentation**
 - **Generate Sample JSON** Files from a JSON Schema
 - **Generate JSON Schema** from a JSON File
 - **XSD to JSON** Schema Converter
 - **Convert JSON Schema** Version



JSON Schema Editor

Design, develop, and edit JSON Schemas in:

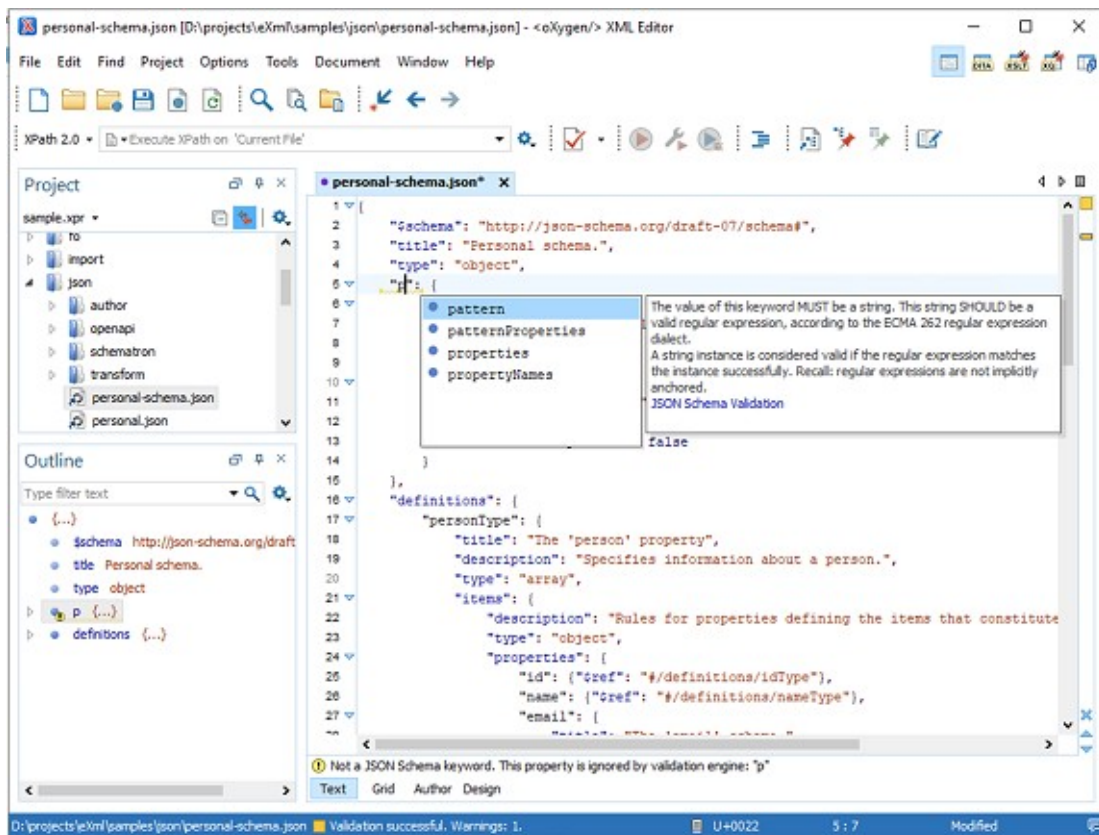
- Text Editing Mode
- Author Editing Mode
- Schema Design Mode



Text Editing Mode

Text editing mode is packed full of editing helpers

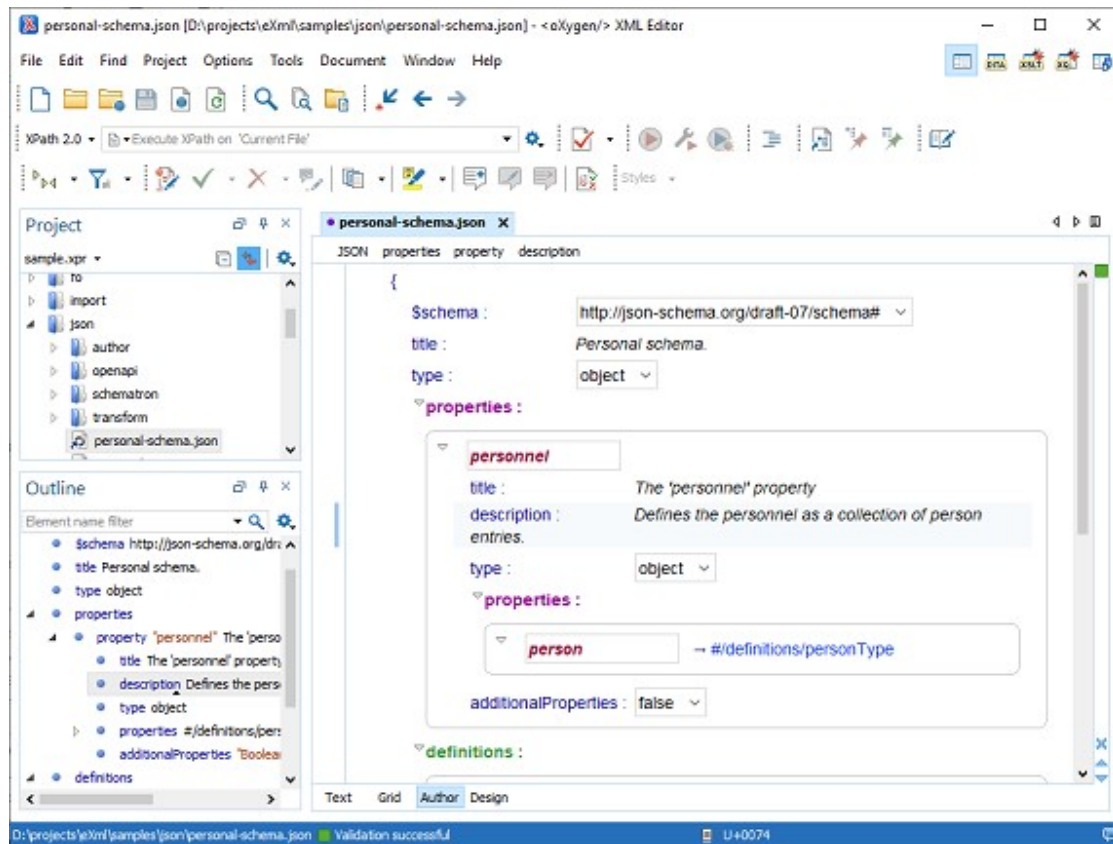
- JSON **Outline View**
- JSON-specific **Syntax Highlighting**
- Search and **Find/Replace**
- **Drag and Drop**
- **Validation**
- **Format and Indent (Pretty Print)**



Author Editing Mode

Visual editing mode for JSON Schema documents:

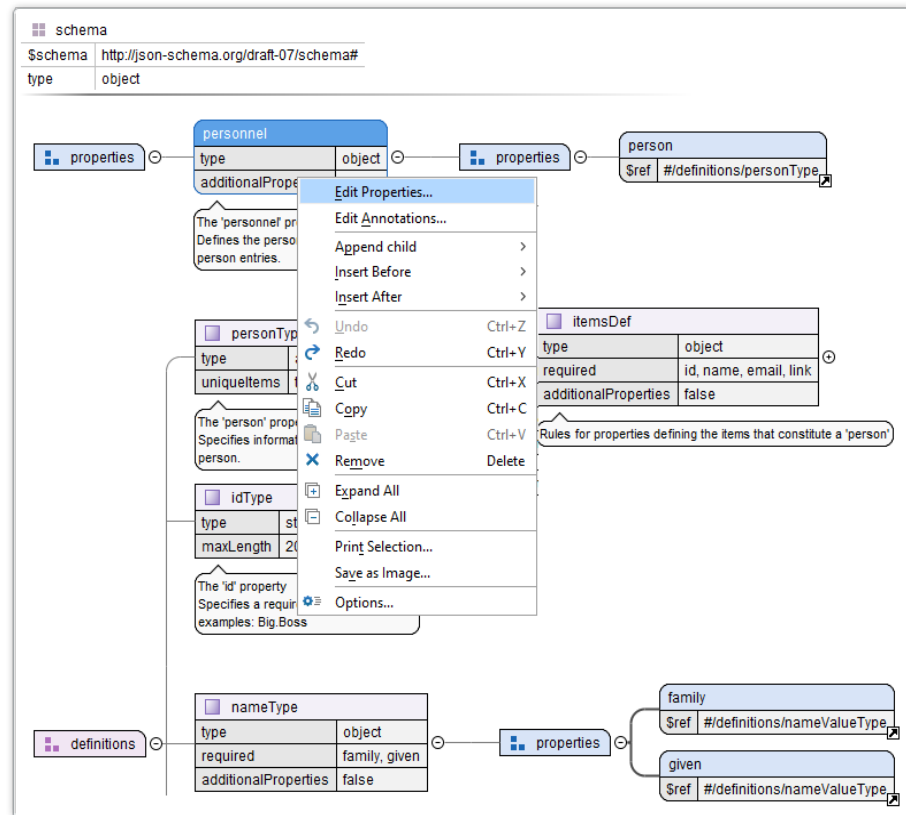
- JSON Schema **framework**
- **Content completion** support
- **Validation**
- Specific **CSS for rendering**
- Create your own **custom JSON framework**



Schema Design Mode

Visualize, edit, and understand JSON Schemas

- **In-Place Component Editing**
- **Drag and drop**
- **Palette view** to Create New Components
- **Schema Refactoring Actions**
- **Print/Save as Image**



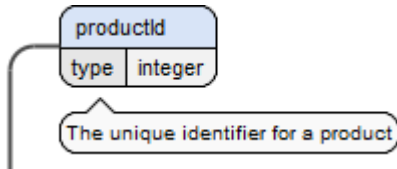
Create JSON Schema from Scratch

- JSON-based product catalog
 - productID – the product identifier
 - productName – the product name
 - price – the selling price of the product
 - tags – an optional set of tags

```
{  
  "productId": 1,  
  "productName": "A green door",  
  "price": 12.50,  
  "tags": ["home", "green"]  
}
```

Create JSON Schema from Scratch

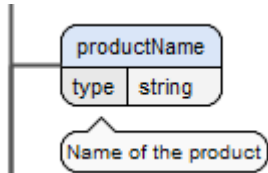
- productID – The identifier for a product
 - Is a numeric value
 - Is required



```
{  
  "productId": 1,  
  "productName": "A green door",  
  "price": 12.50,  
  "tags": ["home", "green"]  
}
```


Create JSON Schema from Scratch

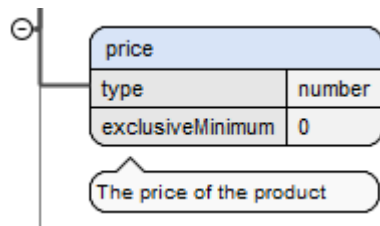
- productName – The name of the product
 - Is a string value
 - Is required



```
{  
  "productId": 1,  
  "productName": "A green door",  
  "price": 12.50,  
  "tags": ["home", "green"]  
}
```

Create JSON Schema from Scratch

- price – The price of the product
 - Is a number value
 - Is required
 - Must be greater than 0

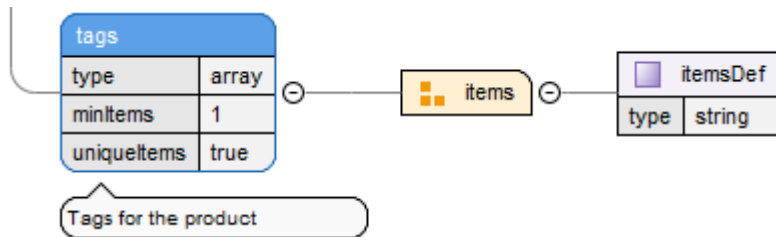


```
{  
  "productId": 1,  
  "productName": "A green door",  
  "price": 12.50,  
  "tags": ["home", "green"]  
}
```

Create JSON Schema from Scratch

- tags – Tags for the product
 - Is an array of string values
 - Is not required
 - At least one item in the array
 - Must be unique relative to one another

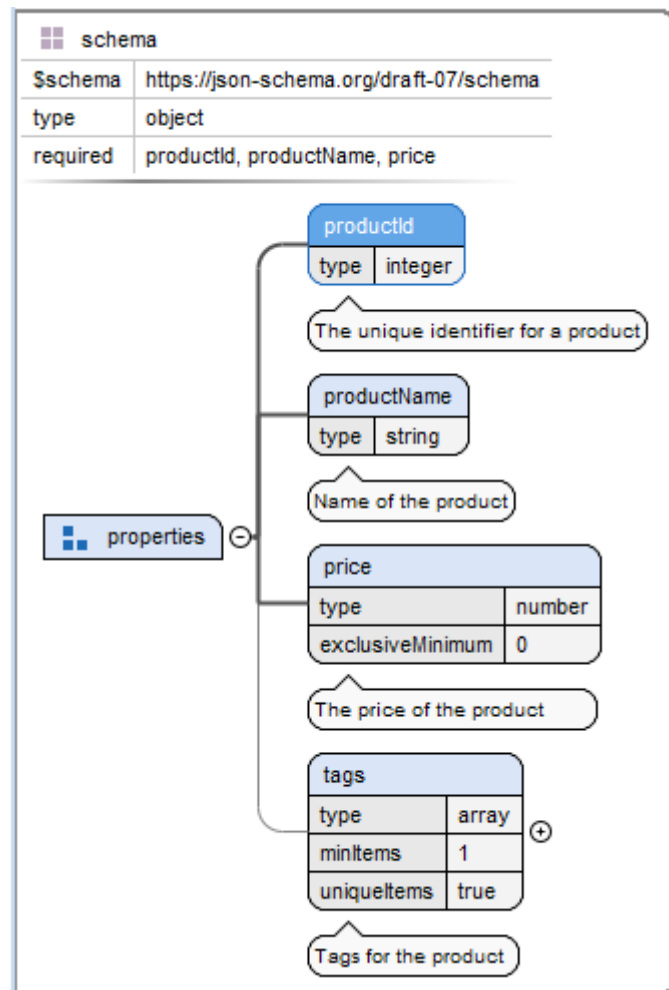
```
{  
  "productId": 1,  
  "productName": "A green door",  
  "price": 12.50,  
  "tags": ["home", "green"]  
}
```



JSON Schema for Product Catalog

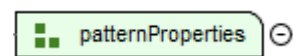
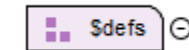
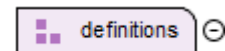
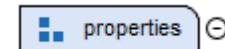
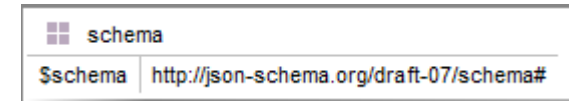
Easy to create a schema from scratch

- Use **document template**
- **Edit** the components **in-place**
- Edit the component **properties** in the in-place **view**
- Add new components using the **append/insert** contextual **actions**
- Change the documentation using the **Annotation dialog** box



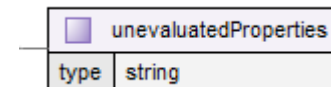
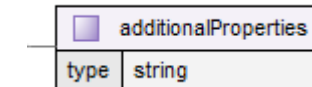
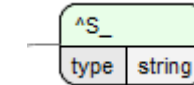
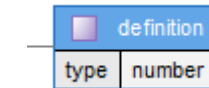
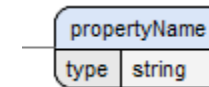
Schema Diagram Components

- **schema** – defines the root schema component
- **properties** – defines a group of *property* components
- **definitions** – contains a group of *definition* components
- **\$defs*** – contains a group of *definition* components
- **pattern properties** – contains a group of *pattern property* components



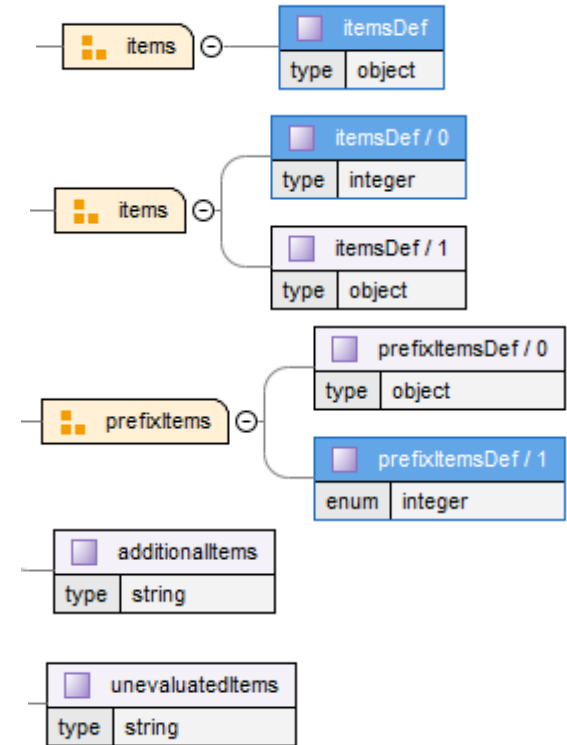
Schema Diagram Components

- **property** – defines a property declaration
- **definition** – contains a declaration of a reusable definition
- **pattern property** – defines a pattern property
- **additional properties** – contains a definition for the additional properties
- **unevaluated properties*** – similar to additional properties except that it can recognize properties declared in subschemas



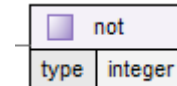
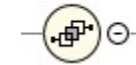
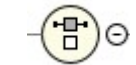
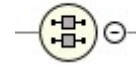
Schema Diagram Components- Arrays

- **Items definition** – definition for all array items
- **Items array** – an array of definitions, one for each item from the array
- **prefixItems*** – an array of definitions, one for each item from the array
- **additional items** – contains a definition for the additional items from an array
- **unevaluated items*** – similar to additional items except that it can recognize properties declared in subschemas



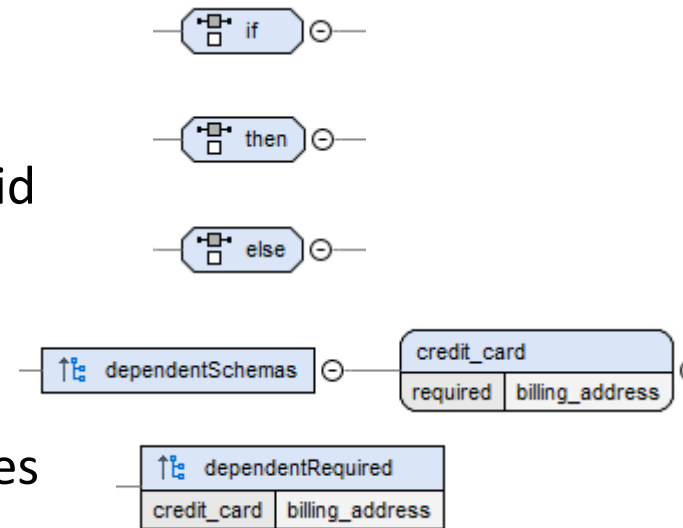
Schema Diagram Components- Composition

- **allOf** – a list of definitions, data must be valid against *all* definitions
- **oneOf** – a list of definitions, data must be valid against exactly *one* of the definitions
- **anyOf** – a list of definitions, data must be valid against *any* definition
- **not** – a definition, data must not be valid against the given definition



Schema Diagram Components- Conditional

- **if** – contains a schema definition for the *if* condition
- **then** – contains a schema definition, data must be valid against it when the *if* condition is *true*
- **else** – contains a schema definition, data must be valid against it when the *if* condition is *false*
- **dependent schemas*** - *conditionally applies a subschema when a given property is present*
- **dependent required*** - *requires that certain properties must be present if a given property is present*

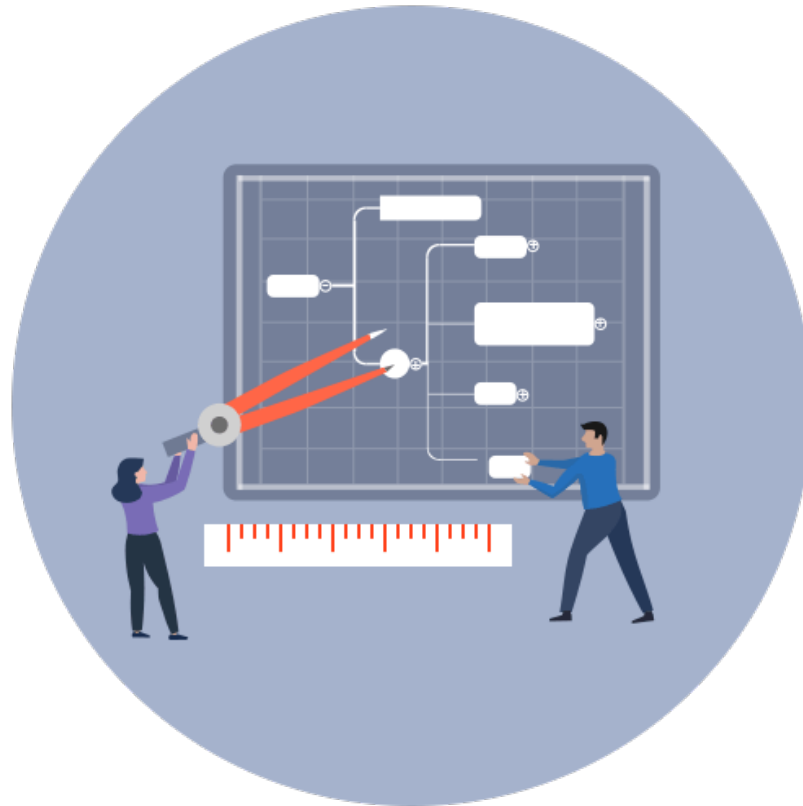


Question: How do you use/intend to use JSON Schema?

- To validate JSON documents
- To define an API
- In a database
- Other (use the Questions pane to provide more details)



Visualize and Edit Complex JSON Schemas



Visualize and Edit Complex JSON Schemas

- Smart navigation
- Zoom in/out
- Expand/Collapse components
- Search/Rename/Go to references
- Go back and forward between components viewed or edited
- Validation markers

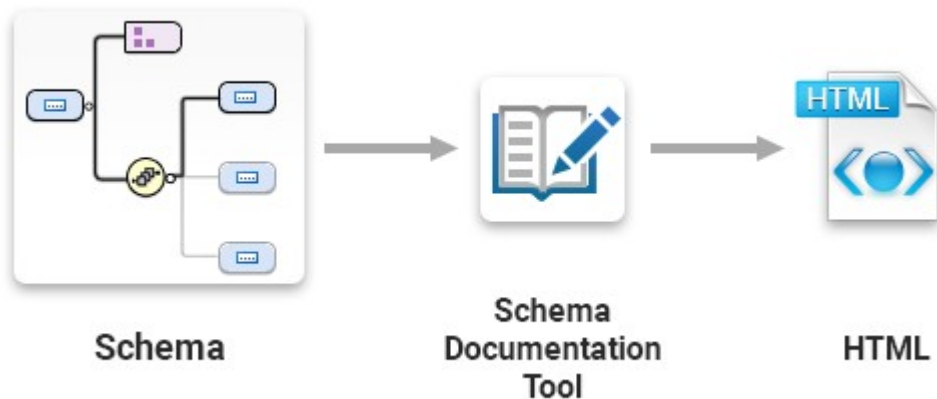
JSON Schema Tools

- Updated to support version 2019-09 and 2020-12
 - Generate JSON Schema Documentation
 - Generate Sample JSON Files from a JSON Schema
 - Generate JSON Schema from a JSON File
 - XSD to JSON Schema Converter
 - Convert JSON Schema Version

oxygenxml.com/ug-editor/topics/json-tools.html

Generate JSON Schema Documentation

- Tool for generating detailed documentation for a JSON Schema file in HTML format



JSON Schema Documentation

- Generate documentation in one file or split into multiple files
- Option to include components details
- Display the diagram image for each component

The screenshot displays a web browser window titled "Documentation for UBL-Order-2". The browser address bar shows the file path: "D:/Projects/VideoDemonstrations/JSON_Schema_Design_v24/JSON_Schema_Design/Samples/json-schem...". The main content area is titled "Property Order" and contains a detailed JSON Schema definition for the "Order" type.

Property Order

Annotations

Description: A document used to order goods and services.

Diagram

The diagram illustrates the structure of the "Order" type. It is represented as a box labeled "Order" with properties: "maxItems": 1, "minItems": 1, and "type": array. A line connects the "Order" box to a box labeled "items", which is connected to a box labeled "ItemsDef". The "ItemsDef" box has a "Sref" property with the value "#/definitions/Order".

Type

array

Constraints

Unique Items: false

Array Items

Items
[UBL-Order-2.1.json.html#/properties/Order/items](#)

Additional Items

true

Used by

Schema
[#/schema](#)

Source

```

"Order": {
  "type": "array",
  "minItems": 1,
  "maxItems": 1,
  "description": "A document used to order goods and services.",
  "items": {"$ref": "#/definitions/Order"}
}
    
```

Showing:

- Annotations
- Diagram
- Properties
- Constraints
- Used By
- Source

Close

OpenAPI/AsyncAPI/JSON-LD

- **OpenAPI** uses JSON Schema to describe data objects for both requests and responses
- **AsyncAPI** schema object is a superset of the JSON Schema Specification Draft 07
- **JSON-LD** uses JSON Schema to define the vocabulary, to validate and annotate JSON-LD documents
- **JSON Schema** can be use to define **any custom vocabulary**



Batch Validate JSON and JSON Schema

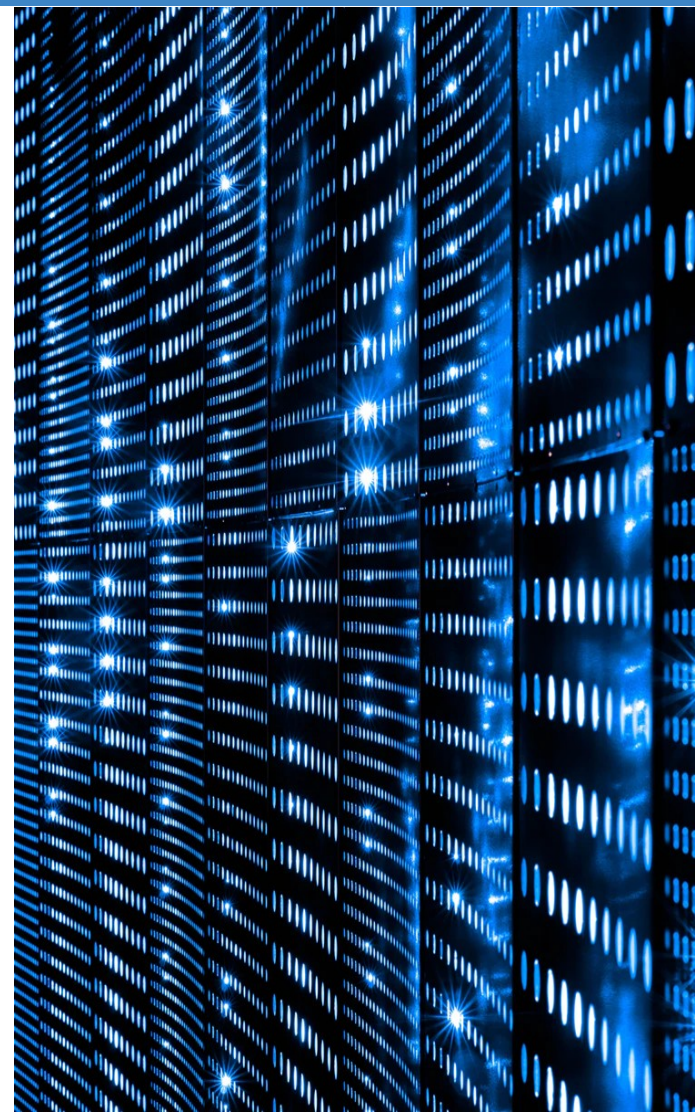
- Use the Validate action from project



- Use the **Oxygen Scripting** from CI/CD



oxygenxml.com/doc/ug-editor/topics/scripting_oxygen_validate.html



Conclusion

- Complete support for JSON Schemas
- JSON Schema Editor
- Validate JSON with JSON Schema
- Editing based on JSON Schema
- Useful JSON Schema Tools



Future Plans

- Improve JSON Schema **Diagram**
- Improve JSON Schema **Text Page**
- Full Support JSON Schema **2020-12**
- Improve **OpenAPI/AsyncAPI** Support
- **Flatten** JSON Schema
- Content Completion for **YAML**



Question: What features are the most important for you?

- ❑ Improve JSON Schema Diagram
- ❑ Improve JSON Schema Text Page
- ❑ Full Support JSON Schema 2020-12
- ❑ Better OpenAPI/AsyncAPI Support
- ❑ Other (use the Questions pane to provide more details)



Resources

- oxygenxml.com/json_schema_editor.html
- oxygenxml.com/doc/ug-editor/topics/editing-JSON-schema.html
- <https://json-schema.org/>
- <https://www.openapis.org>
- <https://www.asyncapi.com>



Video Demos

- oxygenxml.com/demo/json_schema_palette.html
- oxygenxml.com/demo/introducing_the_json_schema_design.html
- oxygenxml.com/demo/json_author.html
- oxygenxml.com/demo/json_tools.html
- oxygenxml.com/demo/json_validation.html
- oxygenxml.com/demo/json_editing.html
- oxygenxml.com/demo/json_query.html



Questions?

Octavian Nadolu
Product Manager at Syncro Soft

octavian.nadolu@oxygenxml.com

Twitter: [@OctavianNadolu](https://twitter.com/OctavianNadolu)

LinkedIn: [octaviannadolu](https://www.linkedin.com/in/octaviannadolu)