ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

JAVASCRIPT OBJECT NOTATION

LIKE,SHARE, FOLLOW ! @esilv_paris | esilv.fr

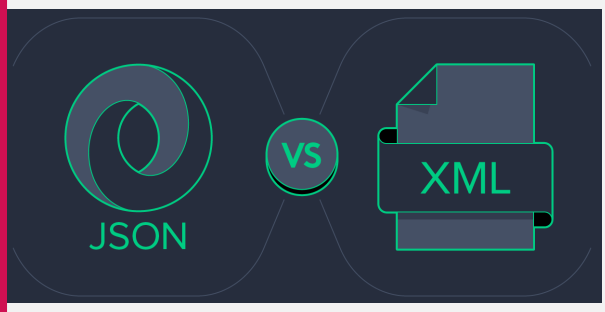
nicolas.travers@devinci.fr

SEMI-STRUCTURED DATA - JSON
JAVASCRIPT OBJECT NOTATION

{JSON}

SWAPI
The Star Wars API

LIKE,SHARE, FOLLOW ! @esilv_paris | esilv.fr
Nicolas.travers@devinci.fr



JSON: JAVASCRIPT OBJECT NOTATION

- XML initially designed for standardizing computer communications
  - Too verbose & space greedy
  - Dedicated to *Web Services*

vs

- JSON (JavaScript Object Notation)
  - Designed for web server to web browser communications
  - Lightweight, text oriented, language independent
  - Used for APIs (Google API, Twitter API) or dynamic web (Ajax)

JSON: (TOO) SIMPLE GRAMMAR

J ::=	$B \mid O \mid A$	JSON expressions
B::=	null|true|false|n|s	Basic values
	$n \in$ Num,$s \in$ Str	
O ::=	$\{l_1 : J_1,\ldots,l_n : J_n\}$	Objects
	$n \geq 0$, $i \neq j \Rightarrow l_i \neq l_j$	
A::=	$[J_1,\ldots,J_n]$	Arrays
	$n \geq 0$	

JSON: KEY-VALUES & TYPING

Key + Value
- "lastname" : "Travers"
- Keys with **quotations**

Identifiers
- **" _id "** commonly used
- Overwrite already stored ids
- Can be automaticaly generated
  - Ex MongoDB: "_id" : ObjectId(1234567890)

Objects/Documents
- Collection of **key/values – unique keys**

```
{ "_id" : 1234,
  "lastname" : "Travers",
  "firstname" : "Nicolas",
  "kind" : 1}
```

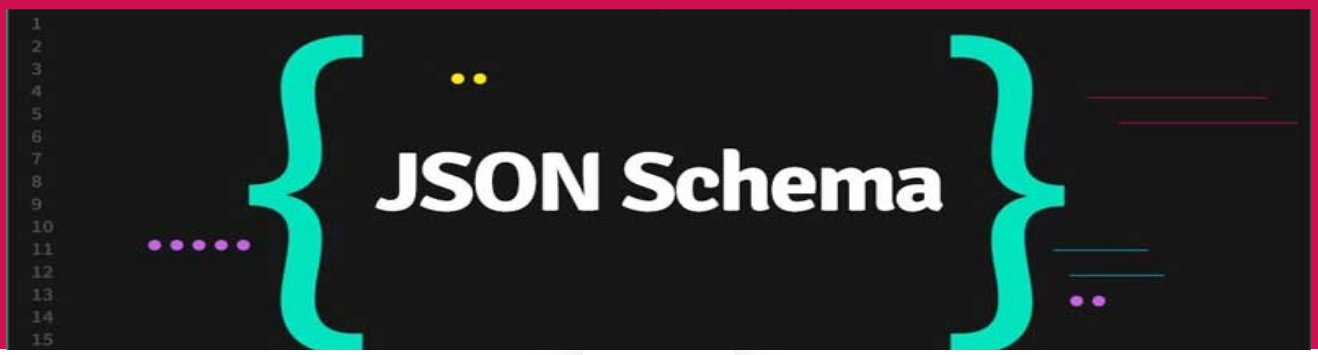
Scalar
- String, Integer, float, boolean, null…

Documents
- Objects **{…}**

List
- Arrays **[…]**
- No typing
  - "lessons" : **[** "SQL", 1, 4.2, null, "NoSQL" **]**
- Can nest documents
  - "doc" : **[**{"test" : 1}**,**
    {"test" : {"nesting" : 1.0}}**,**
    {"key" : "text", "value" : null} **]**

JSON: COMPLETE EXAMPLE

```
{
        "_id" : 1234,
        "lastname" : "Travers", "firstname" : "Nicolas",
        "employers" : [
                { "company" : "ESILV", "starting_date" : "2018-09",
                  "location" : {"street" : "12 avenue Léonard de Vinci","city" : "Paris La Défense", "zip" : 92916}},
                { "company" : "CNAM", "starting_date" : "2007-09", "end_date" : "2018-08",
                  "location" : {"street" : "2 rue Conté","city" : "Paris", "zip" : 75141}},
                { "company" : "UCP", "starting_date" : "2006-09", "end_date" : "2007-08",
                  "location" : {"street" : "2 rue Conté","city" : "Cergy-Pontoise", "zip" : 91000}},
                { "company" : "UVSQ", "starting_date" : "2004-01", "end_date" : "2006-08",
                  "location" : {"street" : "45 avenue des Etats-Unis","city" : "Versailles", "zip" : 78000}}    ],
        "fields" : [ "DB" , "DB optimization", "XML", "NoSQL", "IR" ],
        "hobbies" : ["Star Wars", "BZH"]
}
```



ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

JSON SCHEMA

JSON SCHEMA

- Needs a Schema to validate documents' structure
- Automatic check (exists, inclusion, syntax, queries)

- Requires a typed grammar
  - Objects, arrays,
  - Mandatory/optional keys

http://json-schema.org/

JSON SCHEMA : OBJECTS

```
{
        "type" : "object",
        "properties" : {
                "key1" : {"type": ... },
                "key2" : {"type": ... }
        }
}
```

Object's keys

Types of values
"number" (option – "min", "max")
"string" (option - "pattern")
"anyOf"
"not"

Other options : "definitions" + "$ref"

JSON SCHEMA : KEYS

```
{
        "type" : "object",
        "properties" : {
                "key1" : {"type": ... },
                "key2" : {"type": ... }
        },
        "required" : ["key1"],
        "additionalProperties" : false
}
```

Mandatory keys
Missing = optional

Forbids additional keys

Other options : "minProperties", "maxProperties", "patternProperties"

WARNING! NEGATION

```
{
        "type": "object",
        "properties": {
                "key1": { "type": "integer" }
        },
        "not": { "required": [ "key1" ] }
}
```

JSON SCHEMA : LISTS

```
"arrayKey" : {
        "type" : "array",
        "items" : [ {"type" : ... } ],
        "minItems" : 1,
        "maxItems" : 10
}
```

Other options
"uniqueItems"
"additionalItems"

JSON SCHEMA : EXAMPLE

```
{ "type": "object",
  "properties": {
    "_id": { "type": "integer" },
    "lastname": { "type": "string" },
    "firstname": { "type": "string" },
    "employers": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "company": { "type": "string" },
          "starting_date": { "type": "string" },
          "end_date": { "type": "string" } },
...
          "location": {
            "type": "object",
            "properties": {
              "street": { "type": "string" },
              "city": { "type": "string" },
              "zip": { "type": "integer" } },
            "required": [ "street", "city", "zip" ] },
          "required": [ "company", "starting_date", "location" ] } },
    "fields": { "type": "array", "items": { "type": "string" } },
    "hobbies": {
        "type": "array",
        "items": { "type": "string" } } },
    "required": [ "_id", "lastname", "firstname", "employers",
                  "fields", "hobbies" ]
}
```


JSON SCHEMA: MISCELLANEOUS

PROGRAMMING LANGUAGE-ORIENTED SPECIFICATION

JOI

https://github.com/hapijs/joi

SCHEMA EXTRACTION

https://www.liquid-technologies.com/online-json-to-schema-converter

https://app.quicktype.io/#l=schema

SCHEMA VALIDATION

https://www.jsonschemavalidator.net/

JSON EXAMPLES GENERATOR

From a JSON Schema

https://jsonschematool.ew.r.appspot.com/