IL LINGUAGGIO JSON PER LA RAPPRESENTAZIONE DEI DATI

Per l'interscambio dei dati fra applicazioni, oltre ad XML esiste un altro formato, che ad oggi è il più +utilizzato, chiamato JSON (Java Script Object Notation). Il nome JSON deriva dal fatto che la sintassi utilizzata è analoga a quella di Java Script, JSON è comunque indipendente dal linguaggio JavaScript.

Il linguaggio è stato ideato da Duglas Crockford nei primi anni 2000 per lo scambio di dati fra lato server e lato client di un'applicazione web. Si è diffuso molto rapidamente perché il parsing di dati JSON è immediato in JavaScript, e perché è meno verboso di XML. Comunque ad oggi esistono librerie per il parsing e la creazione di documenti JSON praticamente per tutti i linguaggi di programmazione più diffusi. Il parsing è l'operazione con cui un software "interpreta" i dati ricevuti in JSON (oppure in XML, CSV o in altro formato) per poi utilizzarli al suo interno (ad esempio per istanziare oggetti, salvare i dati ricevuti su un database ecc..). Il software che realizza il parsing è chiamato parser e può essere scritto in diversi linguaggi di programmazione (ad esempio Java, Javascript, Pyton, ecc)

SINTASSI E TIPI DI DATO

JSON è case sensitive.

Lo standard non prevede i commenti poiché alcuni parser potrebbero non riconscerli

Un documento JSON ha estensione **.json.**

JSON consente di rappresentare **oggetti** (un **oggetto** JSON equivale ad un **elemento** XML).

Ogni oggetto è compreso fra due **parentesi graffe** "**{**…**}**" ed è costituito da una lista di **proprietà**. Una proprietà è una coppia **nome: valore** sparate da **due punti. Le proprietà** sono fra loro separate da una **virgola.**

Ogni proprietà è dunque costituita da due elementi, il nome e il valore, e la sintassi è la seguente

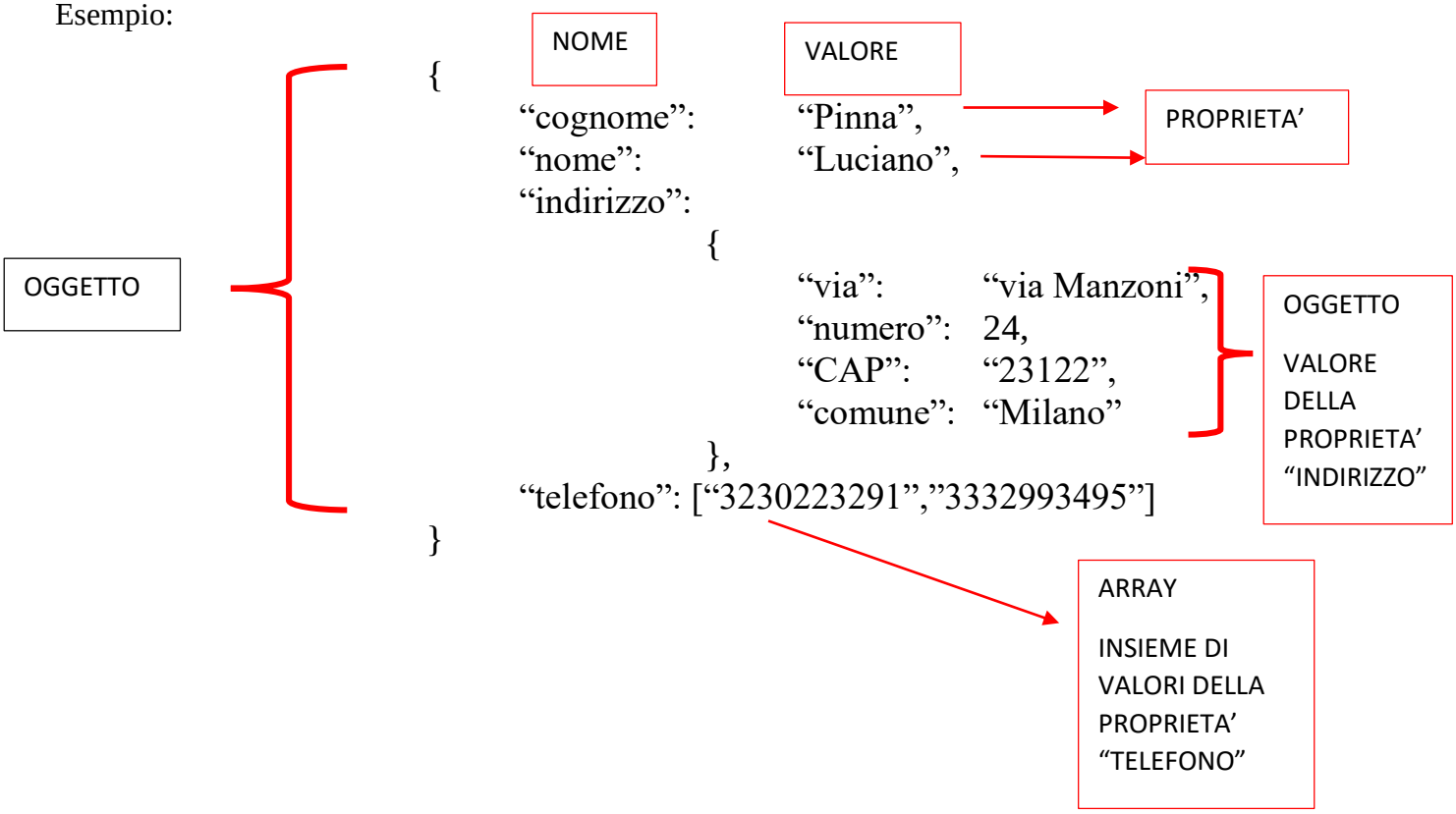
"nome" : "valore"

Il **nome** va sempre messo fra virgolette.

Il **valore** può essere:

- un tipo di dato primitivo. In tal caso se è una stringa va messo fra virgolette, se un numero o un valore booleano non va messo fra virgolette
- Un oggetto. In tal caso, come ogni oggetto, va messo fra parentesi graffe.
- Un array, i cui elementi possono essere di tipo di dato nativo, oggetti o altri array. Quando l'oggetto è un array gli elementi vanno messi fra parentesi quadre.

Esempio:



OSSERVAZIONE: la semplicità di JSON è data dal fatto che è completamente definito da poche cose: **tipi di dato primitivi, oggetti, array.**

Quindi un oggetto Json può essere di uno dei seguenti 4 tipi:

- **Number** (qualsiasi numero)
- **boolean** (valori true o false)
- **string**
- **null**
- **object** (a sua volta un oggetto)
- **array**

si noti che non esistono tipi di dato primitivi per le date e per gli orari, tali tipi di dato si rappresentano generalmente come stringhe nello stesso formato previsto per XML (aaaa-mm-gg e hh:mm:ss).

All'interno delle stringhe i caratteri speciali vanno fatti precedere dal backslash "\":

Carattere	Descrizione
\"	Simbolo «"»
\\	Simbolo «\»
\r	*carriage return*
\n	*new line*
\f	*form feed*
\b	*backspace*
\u...	specificazione di un codice Unicode

JSON SCHEMA

Come per l'XML anche per JSON è possibile specificare uno schema che definisce i tipi di dato delle proprietà e contiene le "regole" che l'oggetto JSON (chiamato anche **istanza**) deve rispettare per essere valido rispetto allo schema stesso. Anche uno schema può essere validato a sua volta rispetto uno schema che stabilisce le regole di definizione degli schemi, chiamato "meta-schema". Attualmente (anno 2025) l'ultima versione stabile (le versioni di json schema sono chiamate draft) di JSON Schema è la 2020-1. Le caratteristiche dei Json Schema si possono trovare qui: sul sito https://json-schema.org/.

Un JSON schema è a sua volta un oggetto JSON (file .json) e quindi è costituito da una lista di proprietà, vale a dire da una lista di coppie nome : valore nel quale sono specificate le regole sintattiche, i tipi di dato, le proprietà ecc.utilizzabili in un JSON schema.

Per dettagli e dubbi sul linguaggio JSON si veda il sito: https://json-schema.org/

Come validatore usiamo questo online: https://www.jsonschemavalidator.net/

Le proprietà presenti in un JSON Schema sono le seguenti:

"$schema": "http://json-schema.org/draft/2020-12/schema",

E' un riferimento allo schema che definisce la struttura di un JSON Schema, vale a dire le proprietà che possono essere utilizzate nel JSON Schema. Usiamo questo (ultimo stabile)

"title": " …. ",

"description": " …. ",

Sono due proprietà che descrivono, uno in maniera sintetica e l'altro in maniera estesa, l'oggetto JSON definito con questo schema

"type": "object" (oppure "array"),

Praticamente assume sempre il valore "object" p "array" per indicare che con questo JSON schema sto descrivendo un oggetto o un array.

"properties":

{

"cognome":{"type":"string"},

"nome":{"type":"string"},

"….":{…..},

}

Qua dentro ci metto le "proprietà", vale a dire le coppie "nome":"valore" che costituiscono l'oggetto JSON che sto definendo.

A sua volta il valore è un oggetto, che ha una proprietà "type" che specifica il tipo di oggetto. Il tipo di oggetto può essere:

- un tipo di dato nativo
- un array
- un oggetto

In quest'ultimo caso, a sua volta, l'oggetto avrà un

"type":"object"
"properties": { …….
}

Il JSON schema dell'oggetto visto nell'esempio precedente, escludendo per ora l'array "telefono" che verrà spiegato in seguito, è il seguente:

```
{
        "$schema":   "http://json-schema.org/draft/2020-12/schema",
        "title":        "voce_rubrica",
        "description":"Elemento di una rubrica",
        "type":         "object",
        "properties":
                    {
                        "cognome": {"type":         "string"},
                        "nome": {"type":    "string"},
                        "indirizzo": {"type":         "object",
                                      "properties":
                                                  {
                                                      "via":{"type":"string"},
                                                      "numero":{"type":"integer"},
                                                      "cap":{"type":"string"},
                                                      "comune":{"type":"string"},
                                                  }
                                    }
                    }
}
```

E' a sua volta un oggetto

E' a sua volta un oggetto, volendo si potrebbe definire una description oltre al type

- Se le proprietà non rispettano l'ordine indicato nello schema, l'oggetto JSON è comunque valido rispetto allo schema. Se, ad esempio, nell'oggetto JSON iniziale inverto la posizione delle proprietà "cognome" e "nome", l'oggetto è comunque valido rispetto allo schema precedente

- Se alcune proprietà indicate nello schema non sono presenti nell'oggetto JSON, quest'ultimo è comunque valido rispetto allo schema. Se ad esempio nell'oggetto JSON iniziale elimino la proprietà "cognome", l'oggetto è comunque valido rispetto allo schema precedente

  Per rendere obbligatoria la presenza di alcune proprietà nell'oggetto JSON è necessario aggiungere al JSON schema, **successivamente alla proprietà "properties**", la proprietà "**required**" assegnandole come valore un array contenente il nome delle proprietà che si vuole rendere obbligatorie

  "required": [""cognome","nome"]

- Se l'oggetto JSON contiene ulteriori proprietà rispetto allo schema, esso è comunque valido. Ad esempio l'oggetto iniziale, che contiene la proprietà telefono, è comunque valido rispetto allo schema precedente.

  Per impedire che un oggetto JSON contenente ulteriori proprietà rispetto a quelle indicate nello schema sia comunque valido, è necessario aggiungere allo schema la seguente proprietà con valore false. Se tale proprietà non è specificata, il suo valore di default è true.

  "additionalProperties":"false"

DEFINIZIONE DI UN ARRAY IN UN JSON SCHEMA

Un array in un JSON schema è definito da un oggetto in cui la proprietà "type" assume valore "array". Se non si specifica nient'altro, gli elementi dell'array possono essere di qualsiasi tipo.

Esempio: si vuole aggiungere all'oggetto "voce rubrica" un array "dettagli" che contiene elementi di qualsiasi tipo:

```
{
        "$schema":   "http://json-schema.org/draft/2022-12/schema",
        "title":        "Voce rubrica",
        "description":"Elemento di una rubrica",
        "type":         "object",
        "properties":
                        {
                                "cognome": {"type":         "string"},
                                "nome": {"type":    "string"},
                                "indirizzo": {"type":         "o      bject",
                                                        "properties":
                                                                {
                                                                        "via":{"type":"string"},
                                                                        "numero":{"type":"integer"},
                                                                        "cap":{"type":"string"},
                                                                        "comune":{"type":"string"},
                                                                }
                                        },
                                "dettagli":{"type":"array"}
                        },
        "required":["cognome","nome"],
        "additionalProperties":false
}
```

Aggiungo un array di qualsiasi tipo con un numero qualsiasi di elementi

JSON valido rispetto allo schema

```
{
        "cognome":"Pinna",
        "nome":"Luciano",
        "indirizzo":
                {
                        "via":"Manzoni",
                        "numero":24,
                        "cap":"25100",
                        "comune":"Milano"
                },
        "dettagli":["amico",1988]
}
```

Array dettagli con elementi di qualsiasi tipo

- Se si vuole specificare nel JSON schema **un unico tipo di dato valido per gli elementi dell'array**, è necessario aggiungere alla definizione della proprietà array, una proprietà "**items**" che come valore ha un oggetto la cui proprietà "type" rappresenta il tipo di dato valido.

  Esempio: si vuole che l'array "dettagli" contenga solamente delle stringhe, e si aggiunge un array contenente i numeri di telefono

```
{
    "$schema":  "http://json-schema.org/draft/2020-12/schema",
    "title":    "Voce rubrica",
    "description":      "Elemento di una rubrica",
    "type":     "object",
    "properties":
            {
                "cognome": {"type":     "string"},
                "nome": {"type":    "string"},
                "indirizzo": {"type":       "object",
                              "properties":
                                    {
                                        "via":{"type":"string"},
                                        "numero":{"type":"integer"},
                                        "cap":{"type":"string"},
                                        "comune":{"type":"string"},
                                    }
                    },
                "dettagli":   {
                              "type":"array",
                              "items":{"type":"string"}
                               }
                "telefono":
                              {
                              "type":"array",
                              "items":{"type":"string"}
                              }
            },
    "required":["cognome","nome"],
    "additionalProperties":false
}
```

Aggiungo un array di stringhe con un numero qualsiasi di elementi

JSON valido rispetto allo schema

```
{
    "cognome":"Pinna",
    "nome":"Luciano",
    "indirizzo":
        {
            "via":"Manzoni",
```

```
                "numero":24,
                "cap":"25100",
                "comune":"Milano"
            },
        "dettagli":["amico","conosciuto nel 1988"],
        "telefono":[“2343223409”,”3432112321”]
}
```

Array dettagli con elementi di tipo string

- Se si vuole impostare un numero massimo e un numero minimo di elementi per un array è necessario definire nel JSON schema tali valori come valori della proprietà minItems e maxItems dell’oggetto array

  Esempio: se si vuole che l’oggetto JSON Voce_rubrica abbia almeno un numero di telefono e al massimo due numeri di telefono, l’array “telefono” va ridefinita nel seguente modo nel JSON schema:

```
"telefono":
        {
                "type":"array",
                "items":{"type":"string"},
                "minItems":1,
                "maxItems":2
        }
```

Valori minimo e massimo degli elementi dell’array

  Questo impedirà di validare un oggetto JSON “Voce rubrica” che abbia nell’array “telefono” meno di uno o più di due elementi

Esempio: JSON schema per definire un array di persone

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",

  "title":"persone",
  "description":"array di persone",
  "type":"array",
  "items":
   {
      "type":"object",
      "properties":
        {
                "cognome":{"type":"string"},
                "nome":{"type":"string"}
        },
        "required":["cognome","nome"]
   }
}
```

Oggetto JSON valido:

```
[
  {
    "cognome": "Pinna",
    "nome": "Luciano"
  },
  {
    "cognome": "Nillo",
    "nome": "Pepep"
  }
]
```

Esempio: JSON schema per definire un array di persone ognuna con un array di numeri di telefono (minimo 1 massimo 2 numeri di telefono)

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "persone",
  "description": "array di persone",
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "cognome": {
        "type": "string"
      },
      "nome": {
        "type": "string"
      },
      "telefono": {
        "type": "array",
        "items": {
          "type": "string"
        },
        "minItems": 1,
        "maxItems": 2
      }
    },
    "required": [
      "cognome",
      "nome",
      "telefono"
    ]
  }
}
```

Oggetto JSON valido:

```
[
```

```
 {
  "cognome": "Pinna",
  "nome": "Luciano",
  "telefono":["3233222123"]
 },
 {
  "cognome": "Nillo",
  "nome": "Pepep",
  "telefono":["3233222123","4542342433"]
 }
]
```

RESTRIZIONI SUI TIPI IN UNO SCHEMA JSON SCHEMA

- Valori numerici
  E' possibile limitare ad un intervallo il valore numerico **di una proprietà** o **di un elemento di un array** in fase di definizione del JSON schema aggiungendo, quando si definisce la proprietà o l'elemento, le proprietà **minimum** e **maximum**.

- Lunghezza delle stringhe
  E' possibile definire il valore massimo e minimo della lunghezza del valore di una proprietà di un oggetto o di un elemento di un array che siano di tipo string.
  Per fare questo si impostano le proprietà numeriche **minLength** e **maxLength.**

- Restrizioni enumerative
  Se si vuole limitare ad un elenco di valori possibili il valore che può assumere una proprietà o un elemento di un array è necessario aggiungere alla proprietà o all'array la proprietà **enum** assegnandole come valore l'elenco dei valori possibili.

  Esempio: se si vuole aggiungere all'oggetto JSON "Voce rubrica" la proprietà "sesso" e far sì che tale proprietà possa assumere solamente i valori M ed F, la proprietà deve essere definita nel JSON schema nel seguente modo:

Esempio: riferendoci all'esempio iniziale "Voce rubrica", se si vuole limitare la proprietà "numero" civico ad un valore positivo, il CAP a esattamente 5 cifre, e la lunghezza dei numeri di telefono a esattamente 10 caratteri e al massimo 2 numeri di telefono, inoltre si vuole aggiungere il sesso della persona che può assumere solo i valori "M" o "F". Il Json schema diventa il seguente:

```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "title":"voce rubrica",
    "type":"object",
    "properties":
    {
       "cognome":{"type":"string"},
       "nome":{"type":"string"},
       "sesso":
          {
             "type":"string",
             "enum":["M","F"]
          },
       "indirizzo":
          {      "type":"object",
             "properties":
              {
                 "via":{"type":"string"},
                 "numero":
                    {
                       "type":"number",
```

```
                    "minimum":1
                  },
                "CAP":
                  {
                      "type":"string",
                      "minlength":5,
                      "maxlength":5
                  },
                "comune":{"type":"string"}
              }
          },
          "telefono":
          {
            "type":"array",
             "items":
             {
               "type":"string",
                "minLength":10,
                 "maxLength":10
             },
               "maxItems":2
          }
      },
       "additionalProperties":false,
       "required":["cognome","nome"]

  }
```

Oggetto JSON valido:

```
{
      "cognome":"Pinna",
      "nome":"Luciano",
       "sesso":"M",
         "indirizzo":
         {
              "via":"via x",
           "numero":24,
           "CAP":"25100",
           "comune":"Milano"
         },
         "telefono":["4543483432","6765443232"]
}
```

Esempio Coordinata geografica

Definire un JSON schema che rappresenti una coordinata. La coordinata è un oggetto formato **obbligatoriamente** da 2 elementi:

- Latitudine (oggetto formato **obbligatoriamente** da gradi (min=0,max=90), minuti (min=0, max=59), secondi (min=0, max=59), orientamento (valori possibili N o S)
- Longitudine (oggetto formato **obbligatoriamente** da gradi (min=0,max=180), minuti (min=0, max=59), secondi (min=0, max=59), orientamento (valori possibili E o W)

JSON schema

```
{
        "$schema": "https://json-schema.org/draft/2020-12/schema",
        "title":"coordinata",
   "type":"object",
   "properties":
          {
           "latitudine":
                          {
                  "type":"object",
                  "properties":
                                 {
                        "gradi":
                            {
                              "type":"integer",
                              "minimum":0,
                              "maximum":90,
                            },
                        "minuti":
                            {
                              "type":"integer",
                              "minimum":0,
                              "maximum":59,
                            },
                        "secondi":
                            {
                              "type":"integer",
                              "minimum":0,
                              "maximum":59,
                            },
                        "orientamento":
                                  {
                              "type":"string",
                              "enum":["N","S"]
                            }
                      },
                    "required":["gradi","minuti","secondi","orientamento"]
                 },
             "longitudine":
                          {
```

```
                    "type":"object",
                    "properties":
                                {
                        "gradi":
                            {
                              "type":"integer",
                              "minimum":0,
                              "maximum":90,
                            },
                        "minuti":
                            {
                              "type":"integer",
                              "minimum":0,
                              "maximum":59,
                            },
                        "secondi":
                            {
                              "type":"integer",
                              "minimum":0,
                              "maximum":59,
                            },
                        "orientamento":
                                {
                              "type":"string",
                              "enum":["E","W"]
                            }
                      },
                    "required":["gradi","minuti","secondi","orientamento"]
                }
            },
        "required":["latitudine","longitudine"],
        "additionalProperties":false
}
```

Oggetto JSON valido

```
{
        "latitudine":
          {
            "gradi":77,
            "minuti":10,
            "secondi":4,
            "orientamento":"N"
          },
  "longitudine":
          {
            "gradi":18,
            "minuti":10,
            "secondi":4,
            "orientamento":"E"
```

```
            }
    }
```

Esempio 152 libro TPS Dato Meteorologico (fare)

Definire uno schema JSON per un oggetto che rappresenta un dato meteorologico costituito dai valori numerici interi della temperatura (validi tra -50° e +50°), dell'umidità relativa (validi tra l'1% e il 100%) e della pressione atmosferica (validi tra 800 mbar e 1200 mbar). Definire un oggetto JSON valido per tale dato meteorologico.

Fare esercizi del libro di TPS a p. 135,136, sono gli stessi già svolti in XML, svolgerli in JSON realizzando JSON schema e un oggetto Json valido.

21 giornale: (tranne l'elemento "Intestazione" che è ancora un titolo e quindi non ha senso)

23 Verifica quiz: (punteggio minimo=0, numero di risposte minimo=3, massimo=5)

30 Cassonetti: tranne la parte in Java. Latitudine in decimale è un numero compreso fra 90 e -90 con esattamente 6 cifre decimali. Longitudine è un numero compreso fra -180 e 180 con esattamente 6 cifre decimali.

E gli altri a piacere