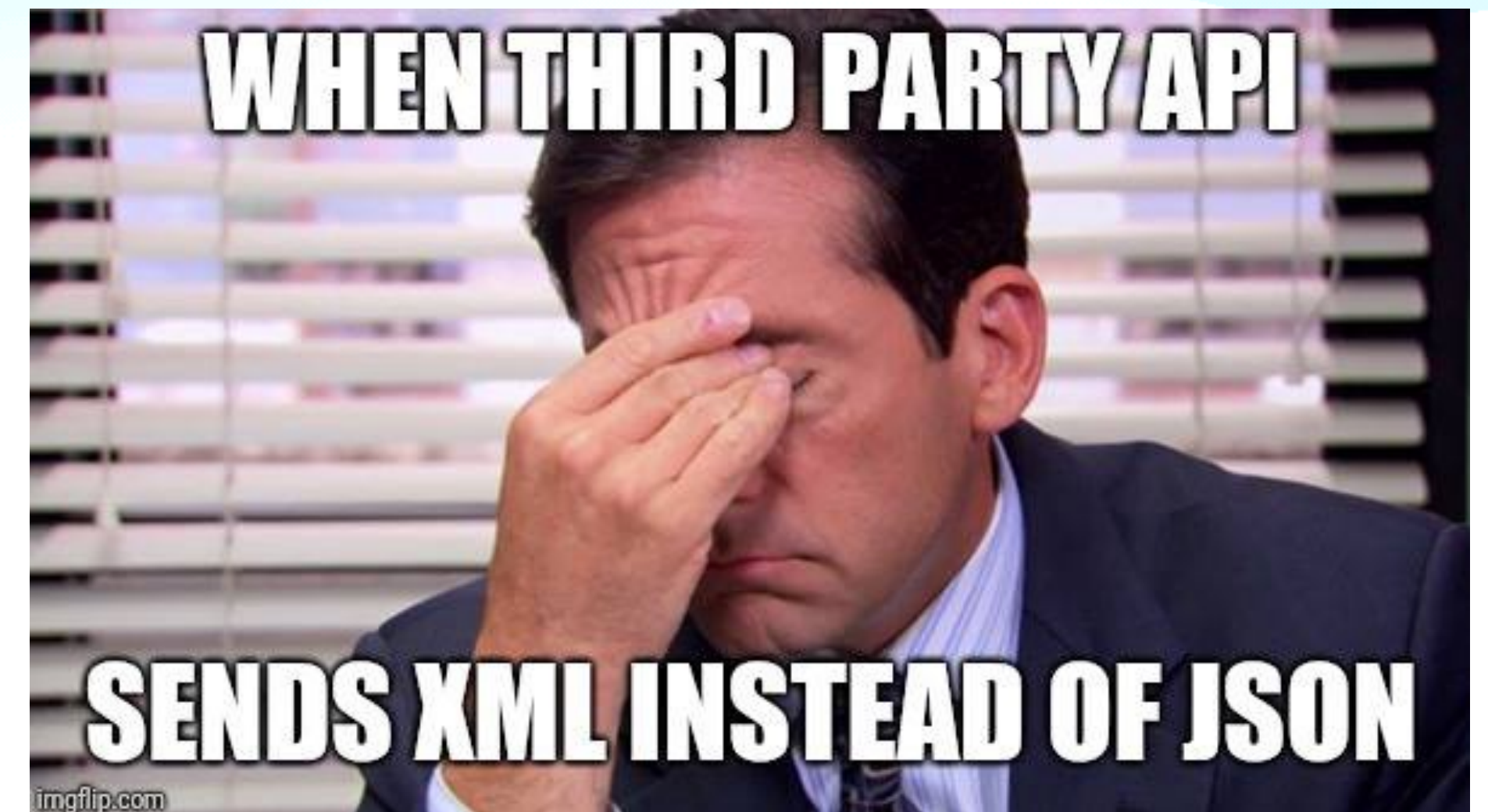


# Data Persistence

## Files and Data Formats

Jorge Edison Lascano  
Object Oriented Programming ESPE





# Data persistence

- Data will last
- Data will stay through time
- Data will persist
- Files
  - Txt, csv, XML, JSON, ...
- Data bases
  - SQL, No-SQL

# Data Persistence: Text Files

## Plain text

- The programmer must organize the information
- The programmer decides what separator is used

- Student: id, name, birthDate, GPA

- student.txt

- 1:Edison Lascano:17/12/1970:9.0
    - 2:Jeremy Cuadrado:06/02/2003:7.5
    - 3:Juan Pinza:15/03/2004:8.0

```
1:Edison Lascano:17/12/1970:9.0
2:Jeremy Cuadrado:06/02/2003:7.5
3:Juan Pinza:15/03/2004:8.0
4:Lady Cajilima:10/02/1999:8.1
```



# Data Persistence: Text Files

## Plain text

- Issues
  - Information can be confused some times
  - The changes in data may be overridden
  - Programming is going to be long and difficult
  - We have to separate each content of every line
  - The separator can be used as part of the data

# Data Persistence: Files

## CSV : Comma Separated Values

- The programmer must use
  - , comma
  - ; semicolon
- Student: id, name, birthDate, GPA
- student.csv
  - 1,Edison Lascano,17/12/1970,9.0
  - 2,Jeremy Cuadrado,06/02/2003,7.5
  - 3,Juan Pinza,15/03/2004,8.0

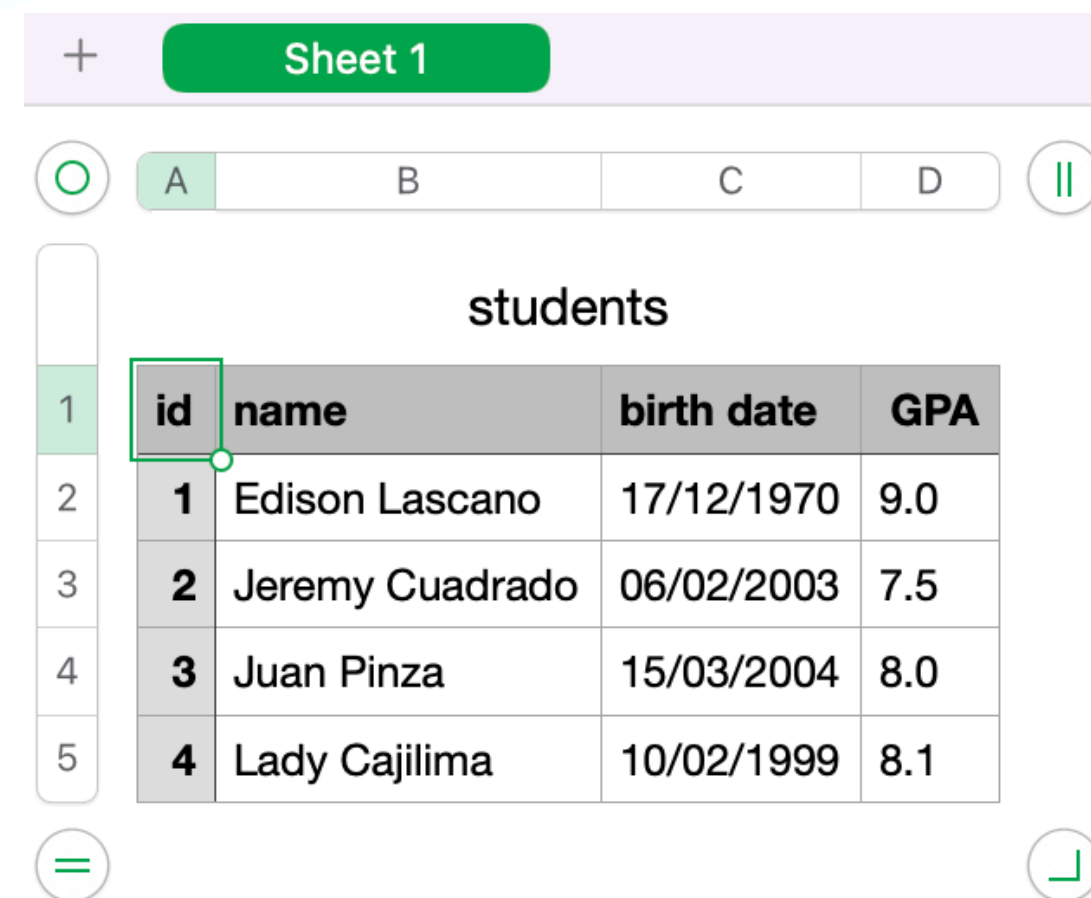
```
1,Edison Lascano,17/12/1970,9.0
2,Jeremy Cuadrado,06/02/2003,7.5
3,Juan Pinza,15/03/2004,8.0
4,Lady Cajilima,10/02/1999,8.1
```



# Data Persistence: Files

## CSV : Comma Separated Values

- The programmer must program everything
- There can be libraries
- These files can be opened using MS Excel or any other spreadsheets program



The image shows a screenshot of a spreadsheet application. At the top, there is a tab labeled "Sheet 1". Below the tab, the spreadsheet grid is visible. The first row is a header row with the following columns: "id", "name", "birth date", and "GPA". The first column is labeled "students" in the first row. The data rows are as follows:

	id	name	birth date	GPA
1	1	Edison Lascano	17/12/1970	9.0
2	2	Jeremy Cuadrado	06/02/2003	7.5
3	3	Juan Pinza	15/03/2004	8.0
4	4	Lady Cajilima	10/02/1999	8.1

# Data Persistence

## XML : eXtensible Markup Language

- Main use of XML files is for configuration, they can also be used for data storage
- Structure of an XML file
  - <tag attribute="value">
    - text (value or data)
  - </tag>
- Student: id, name, birthDate, GPA
- student.xml
  - <student>
  - <id>1</id>
  - <name>Edison Lascano</name>
  - <birthDate>17/12/1970</birthDate>
  - <gpa>9.0</gpa>
  - </student>

```
<students>
  <student>
    <id>1</id>
    <name>Edison Lascano</name>
    <birthDate>17/12/1970</birthDate>
    <gpa>9.0</gpa>
  </student>
  <student>
    <id>2</id>
    <name>Jeremy Cuadrado</name>
    <birthDate>06/02/2003</birthDate>
    <gpa>7.5</gpa>
  </student>
  <student>
    <id>3</id>
    <name>Juan Pinza</name>
    <birthDate>15/03/2004</birthDate>
    <gpa>8.0</gpa>
  </student>
  <student>
    <id>4</id>
    <name>Lady Cajilima</name>
    <birthDate>10/02/1999</birthDate>
    <gpa>8.1</gpa>
  </student>
</students>
"students.xml" 31L, 638B written
```



# Data Persistence

## XML : eXtensible Markup Language

- Advantages
  - Platform independent
  - Data is very well separated, better organization
  - Data specific
  - Human language, more than a computer language
  - Readable, xml can be validated
  - Standard , there are many libraries
- Disadvantages
  - More space is needed, very large files
  - XML syntax is redundant



# Data Persistence

## XML : eXtensible Markup Language

- It can be validated
- 

### XML Validator

[Add to Fav](#) [New](#) [Save & Share](#)

Sample ↺ 📁 💾 ✓ 🗑️ 📄 🔗

```
1 <students>
2 <student>
3   <id>1</id>
4   <name>Edison Lascano</name>
5   <birthDate>17/12/1970</birthDate>
6   <gpa>9.0</gpa>
7 </student>
8 <student>
9   <id>2</id>
10  <name>Jeremy Cuadrado</name>
11  <birthDate>06/02/2003</birthDate>
12  <gpa>7.5</gpa>
13 </student>
14 <student>
15   <id>3</id>
16   <name>Juan Pinza</name>
17   <birthDate>15/03/2004</birthDate>
18   <gpa>8.0</gpa>
19 </student>
20 <student>
21   <id>4</id>
22   <name>Lady Cajilima</name>
23   <birthDate>10/02/1999</birthDate>
```

Ln: 1 Col: 10 size: 534 B T T

☒ Auto [Validate XML](#) [Choose a file..](#) [Load URL](#) [Download](#)

Valid XML

# Data Persistence

## XML : eXtensible Markup Language

- It can be validated
- <https://codebeautify.org/xmlvalidator>

### XML Validator

Add to Fav

New

Save & Share

Sample ↻ 📁 💾 ✓ 🗑️ 📄 🔗

1 <students>

2 <student

3 <id>1</id>

4 <name>Edison Lascano</name>

5 <birthDate>17/12/1970</birthDate>

6 <gpa>9.0</gpa>

7 </student>

8 <student>

9 <id>2</id>

10 <name>Jeremy Cuadrado</name>

11 <birthDate>06/02/2003</birthDate>

12 <gpa>7.5</gpa>

13 </student>

14 <student>

15 <id>3</id>

16 <name>Juan Pinza</name>

17 <birthDate>15/03/2004</birthDate>

18 <gpa>8.0</gpa>

19 </student>

20 <student>

21 <id>4</id>

22 <name>Lady Cajilima</name>

23 <birthDate>10/02/1999</birthDate>

Ln: 2 Col: 9

size: 533 B

☒ Auto

✓ Validate XML

📁 Choose a file..

🔗 Load URL

📄 Download

Error : InvalidAttr  
Line : 3  
Message : boolean attribute '<id>' is not allowed.



# Data Persistence

## JSON : JavaScript Object Notation

- Main use of JSON is for data sharing
- Data format
- Structure of a JSON file

- [{
  - “key”:”value”, { ...} ]

- Student: id, name, birthDate, GPA

- student.json

- {“id”:1, ”name”:”Edison Lascano”, “birthDate”:”17/12/1970”, “GPA”:9.0}

```
{"students": [  
  {"id":1,"name":"Edison Lascano","birthDate":"17/12/1970","gpa":9.0},  
  {"id":2,"name":"Jeremy Cuadrado","birthDate":"06/02/2003","gpa":7.5},  
  {"id":3,"name":"Juan Pinza","birthDate":"15/03/2004","gpa":8.0},  
  {"id":4,"name":"Lady Cajilima","birthDate":"10/02/1999","gpa":8.1}  
]}
```

# Data Persistence

## JSON : JavaScript Object Notation

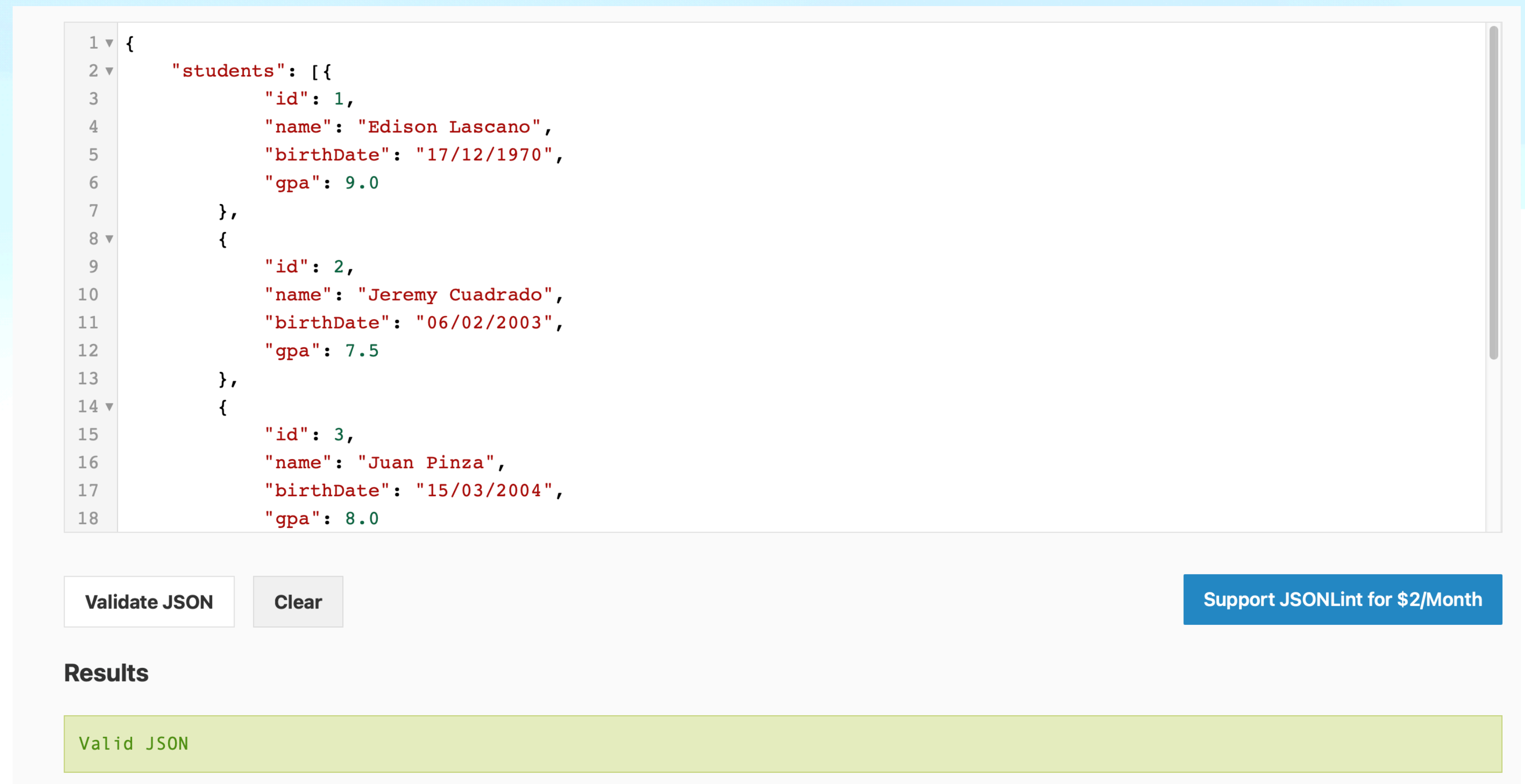
- Lightweight format
- transparent
- Widely used in web applications, mobile apps, and any other client server application
- Understandable
- Readable
- Libraries for any programming language



# Data Persistence

## JSON : JavaScript Object Notation

- It can be validated
- <https://jsonlint.com/>



The screenshot displays the JSONLint validation interface. A text area contains a JSON object with a 'students' array. The array has three elements, each a student object with 'id', 'name', 'birthDate', and 'gpa' properties. Below the text area are 'Validate JSON' and 'Clear' buttons. A blue button on the right says 'Support JSONLint for \$2/Month'. The 'Results' section shows a green bar with the text 'Valid JSON'.

```
1 {  
2   "students": [{  
3     "id": 1,  
4     "name": "Edison Lascano",  
5     "birthDate": "17/12/1970",  
6     "gpa": 9.0  
7   },  
8   {  
9     "id": 2,  
10    "name": "Jeremy Cuadrado",  
11    "birthDate": "06/02/2003",  
12    "gpa": 7.5  
13  },  
14  {  
15    "id": 3,  
16    "name": "Juan Pinza",  
17    "birthDate": "15/03/2004",  
18    "gpa": 8.0  
19  }  
20 }]
```

Validate JSON Clear Support JSONLint for \$2/Month

**Results**

Valid JSON

# Data Persistence

## JSON : JavaScript Object Notation

- It can be validated
- <https://jsonlint.com/>

```
1 {  
2   "students": "id": 1,  
3   "name": "Edison Lascano",  
4   "birthDate": "17/12/1970",  
5   "gpa": 9.0  
6 }, {  
7   "id": 2,  
8   "name": "Jeremy Cuadrado",  
9   "birthDate": "06/02/2003",  
10  "gpa": 7.5  
11 }, {  
12   "id": 3,  
13   "name": "Juan Pinza",  
14   "birthDate": "15/03/2004",  
15   "gpa": 8.0  
16 }, {  
17   "id": 4,  
18   "name": "Lady Cajilima",
```

Validate JSON

Clear

Support JSONLint for \$2/Month

### Results

```
Error: Parse error on line 2:  
{      "students": "id": 1,      "name": "Ediso  
-----^  
Expecting 'EOF', '}', ',', ']', got ':'
```