



JSON

JavaScript Object Notation

- lightweight data-interchange text format
- straightforward to use with JavaScript (\neq Java) but language independent
- popular alternative to XML
 - Similarities:
 - Parseable
 - Human-readable
 - Hierarchical
 - Differences:
 - No end tag
 - Shorter
 - Can use arrays

Syntax

- name (or “key”)/value pairs: name (in double quotes), followed by a colon, followed by a value

```
"first_name":"Bob"
```

- JSON object held in curly braces:

```
{"first_name":"Bob"}
```

- Pairs are separated by commas:

```
{"first_name":"John", "last_name":"Doe"}
```

- Arrays use square brackets:

```
{  
  "students":["Alice", "Bob", "Charly"]  
}
```

Valid data types

- **JSON Strings**, written in double quotes.

```
{"name":"Bob"}
```

- **JSON Numbers** (integer or floating point)

```
{"age":22}
```

- **JSON Booleans** (true/false)

```
{"allergic":true}
```

- **JSON null**

```
{"middlename":null}
```

- **JSON Arrays**

```
{  
  "students":["Alice", "Bob", "Charly"]  
}
```

- **JSON Objects**

```
{  
  "student":{"name":"Bob", "age":22}  
}
```

Example



JSON Files

- file extension: ".json"
- MIME type: "application/json"

JSON and Java

- Multiple libraries: Gson, Jackson, Moshi, javax.json...

```
Gson gson = new Gson();
```

- Parse JSON string as structured text:

```
JsonObject aliceON = gson.fromJson(aliceJsonString, JsonObject.class);
```

- Parse JSON string as objects tree:

```
Student alice = gson.fromJson(aliceJsonString, Student.class);
```

- Serialize java object to JSON string:

```
Student bob = new Student("Bob", 22);
```

```
String bobJsonString = gson.toJson(bob);
```

JSON and Python

- `json` module with `loads` and `load` functions to build a dictionary from (resp.) a string or a file