Data Representation

Marco Torchiano

Version 1.0.0 - May 2020

License 

Open Data

Open Data

Data that can be:

- Used
- Reused
- Redistributed

by whoever for whatever purpose.

> *Knowledge is open if anyone is free to access, use, modify, and share it — subject, at most, to measures that preserve provenance and openness.*
>
> *http://opendefinition.org*

Open Licenses (must)

- Use:
  - allow free use of the licensed work.
- Redistribution:
  - allow redistribution of the licensed work, including sale, whether on its own or as part of a collection made from works from different sources.
- Modification:
  - allow the creation of derivatives of the licensed work and allow the distribution of such derivatives under the same terms of the original licensed work.

Open Licenses (must)

- Separation:
  - allow any part of the work to be freely used, distributed, or modified separately from any other part of the work or from any collection of works in which it was originally distributed.
  - All parties who receive any distribution of any part of a work within the terms of the original license should have the same rights as those that are granted in conjunction with the original work.

Open Data

- Online
- Machine Readable
- Open Format
    - CSV
    - JSON
    - XML

See also: 5 ★ OPEN DATA

Data representation

Tabular Data

Tables are structured into *rows*

- a row contains information about some item
- all rows consist of the same number of *cells* (possibly empty)
- cells in the same position describe the same property of the items
- the first row contains the *headers*, that identify the name of the properties

Tabular data example

ID	Surname	Name
4321	Snow	Jon
5765	Lannister	Tyrion
4663	Targaryen	Daenerys
9896	Stark	Arya

Tabular Data in Javascript

It is *usually* represented using an array

- each element is a row
- row is an object with (named) properties representing the cells

```
[{ID:4321,Surname:"Snow",Name:"Jon"},
{ID:5765,Surname:"Lannister",Name:"Tyrion"},
{ID:4663,Surname:"Targaryen",Name:"Daenerys"},
{ID:9896,Surname:"Stark",Name:"Arya"}]
```


XML

Extensible Markup Language

- Markup language (like HTML but with stricter rules)
- Textual format both human and machine readable
- Validation of structure:
  - Document Type Definition (DTD)
  - XML Schema
- Available parsers for most languages
- XML DOM
- mime type is `application/xml`

Represent a table in XML

```
<characters>
  <character><id>4321</id><surname>Snow</surname>
             <name>Jon</name>
  </character>
  <character><id>5765</id><surname>Lannister</surname
             <name>Tyrion</name>
  </character>
  <character><id>4663</id><surname>Targaryen</surname
             <name>Daenerys</name>
  </character>
  <character><id>9896</id><surname>Stark</surname>
             <name>Arya</name>
  </character>
</characters>
```


Reading XML

Parsing can be perfomed with an object of `DOMParser` type

- method `parseFromString()`

```
var parser = new DOMParser();
var doc = parser.parseFromString(xmlContent,
                                 "application/xml");
var arya = doc.getElementsByTagName("name")[3];
console.log(arya.textContent);
```

↪ `Arya`

CSV

Comma Separated Values

Textual representation of tabular data

- text file containing records separated by newlines
- fields inside a record separated by commas `,`
- first record may contain *headers*
- double-quote (`"`) are used to enclose text
  - ddq: `"This ""is"" quoted"`
- mime type is `text/csv`

Represent a table in CSV

```
ID,Surname,Name
4321,Snow,Jon
5765,Lannister,Tyrion
4663,Targaryen,Daenerys
9896,Stark,Arya
```

Reading a CSV

```
function naifCSVParser(csv) {
  var rows = csv.split(/\n|\r/);
  var fields = rows[0].split(",");
  var data = [];
  for(var i=1; i<rows.length; ++i){
    var row = {};
    var cells = rows[i].split("\,");
    for(var j=0; j<fields.length; ++j){
      row[fields[j]] = cells[j];
    }
    data.push(row);
  }
  return data;
}
```


Reading CSV

```
var jd = naifCSVParser(csvContent);
console.log(jd[3].Name);
```

↪ `Arya`

JSON

Java Script Object Notation

Format for data interchange:

- lightweight format
  - more compact than XML
- language independent
- “self-describing” and easy to understand
- standard
  - The file extension is `.json`
  - The MIME type is `application/json`

JSON Table representation

Property names must be quoted within `"`

```
[
{"ID":4321,"Surname":"Snow","Name":"Jon"},
{"ID":5765,"Surname":"Lannister","Name":"Tyrion"},
{"ID":4663,"Surname":"Targaryen","Name":"Daenerys"},
{"ID":9896,"Surname":"Stark","Name":"Arya"}
]
```

Reading JSON

Parsing can be perfomed with the `JSON` object

- method `parse()`
  - is safe against malicious code,
  - i.e. you cannot insert a call as in `eval()`

```
var jd = JSON.parse(jsonContent);
console.log(jd[3].Name);
```

↪ `Arya`

References

- W3C. Model for Tabular Data and Metadata on the Web. http://www.w3.org/TR/tabular-data-model/
- W3C. Extensible Markup Language (XML) 1.0. https://www.w3.org/TR/REC-xml/
- W3C. CSV on the Web: A Primer. https://www.w3.org/TR/2016/NOTE-tabular-data-primer-20160225/
- IETF. Common Format and MIME Type for Comma-Separated Values (CSV) Files. https://www.ietf.org/rfc/rfc4180.txt
- T. Bray, Ed. The JavaScript Object Notation (JSON) Data Interchange Format. https://tools.ietf.org/html/rfc7159
- W3Schools. JSON Tutorial. http://www.w3schools.com/json/default.asp
- D. Crockford, The application/json Media Type for JavaScript Object Notation (JSON), IETF. https://tools.ietf.org/html/rfc4627