college CDI

Data processing technologies (TTD)

Class 04

JSON

Standing for *JavaScript Object Notation*, JSON is a data format making it easier to store informations and allowing data exchanges between any programming languages. Although JSON was created independently from the EXMAScript specification, it is now linked to JavaScript which includes a JSON object. Although it isn't considered as a programming language, it is important for a web developer to learn it and developers almost consider JSON as part of JavaScript nowadays.

JSON is an alternative to the very well known XML. Testing your files often require them to be hosted or to use a virtual server such as MAMP.

Pros and cons

Almost flawless, you will soon find out this language is very easy to read, understand and use.

Pros

The format can be understood by both humans and machines. It doesn't require any real learning except for the syntax using a specific punctuation.

It doesn't depend on any other language (open data exchange) and it is taken in charge by several languages : JavaScript, PHP, Perl, Python, Ruby, Java, etc.

It allows to stock different types of data : strings (including base64 images), numbers, arrays, objects, booleans, null, etc.

It's tree structure and simple syntax make it light and efficient.

It is widely used to integrate different types of contents to web pages, such as APIs.

Cons

Just like for any database methods, security measures need to be put in place to protect confidential informations.

The fact that the syntax is very simple may sometimes pause problems. For instance, JSON use no tags such as XML so the developer needs to know the data structure.

JSON vs. XML

JSON	XML
Object has a type	Data is typeless
Types: string, number, array, Boolean	All data should be string
Data is readily accessible as JSON objects	XML data needs to be parsed.
Supported by most browsers.	Cross-browser XML parsing can be tricky
Has no display capabilities.	XML is a markup language.
Supports only text and number data type.	Also support various types such as images, charts...
Retrieving value is easy	Retrieving value is difficult
Supported by many Ajax toolkit	Not fully supported by Ajax toolkit
Fully automated way of deserializing/serializing JavaScript.	JavaScript code needed to serialize/deserialize from XML
Native support for object.	The object has to be express by conventions mostly missed use of attributes and elements.
It supports only UTF-8 encoding.	It supports various encoding.
It doesn't support comments.	It supports comments.
JSON files are easy to read as compared to XML.	XML documents are relatively more difficult to read and interpret.
It does not provide any support for namespaces.	It supports namespaces.
It is less secured.	It is more secure than JSON.

Keys and Values

JSON data consists in pairs of keys (equivalent of properties) and values placed between double quote marks and separated by colons in an object placed between braces. Together they make key/value pairs of an object. Elements of JSON's objects can be of various types : strings, numbers, arrays, objects, booleans, null, etc. JSON files must be encoded in UTF-8 (or UTF-16 / UTF-32) and save with .json extension.

Key: A key is always a string enclosed in quotation marks.

Value: A value can be a string, number, boolean expression, array, or object.

```
"car" : "Mazda"
```

Structure

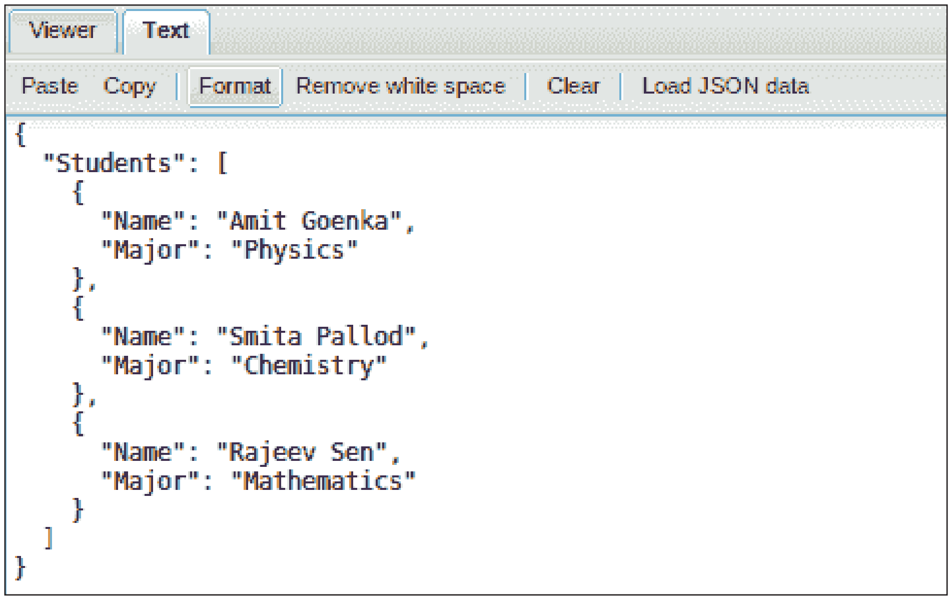
{...}	Brace brackets are used to define an object.
[...]	Brackets are used to define an array.
,	Commas are used to separate elements of an object (no comma after the last element or the element will not be valid).

```
{                                                  // Start of the object
        "Course name" : "LS2",                     // String element
        "Topic" : "Script languages",
        "Students" : [                             // Start of array element
                {                                  // Start of array's object
                        "Last name" : "Norris",
                        "First name" : "Chuck",
                        "Age" : 75,                // Numbers don't require double quotes
                        "Country" : "USA"
                },                                 // End of array object
                {
                        "Last name" : "Doe",
                        "First name" : "John",
                        "Age" : 44,
                        "Country" : "UK"
                },
                {
                        "Last name" : "The Poo",
                        "First name" : "Winnie",
                        "Age" : 10,
                        "Country" : "FRANCE"
                }
        ]                                          // End of array element
}                                                  // End of the object
```

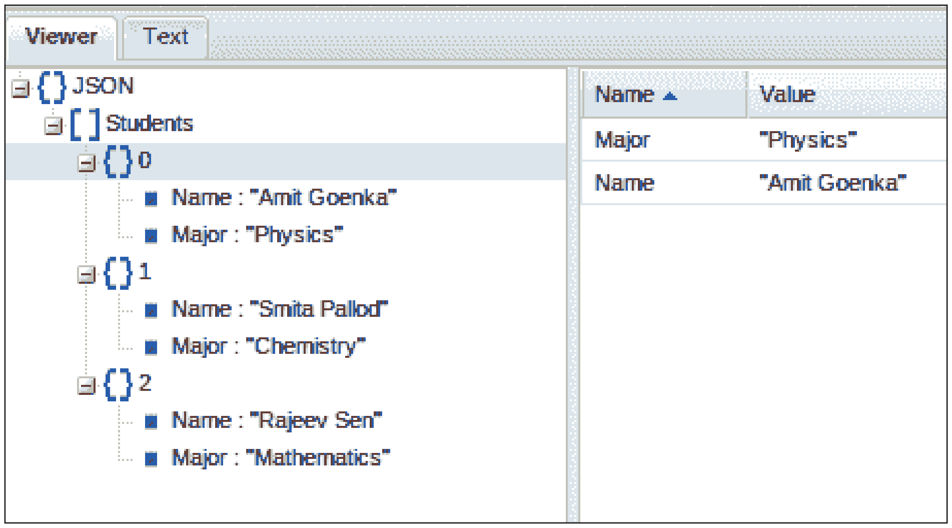
Viewing and validating

Especially at the beginning, it is useful to be able to visualize and validate the source code. Many online editors / validators are available for you to use such as the very well known *jsonviewer.stack.hu*.

Al there is to do is to copy and paste your code in the the application's "Text" window. The validator the indicates if errors are found and the viewer window shows the tree structure of the code.



```
{
  "Students": [
    {
      "Name": "Amit Goenka",
      "Major": "Physics"
    },
    {
      "Name": "Smita Pallod",
      "Major": "Chemistry"
    },
    {
      "Name": "Rajeev Sen",
      "Major": "Mathematics"
    }
  ]
}
```



```
JSON
  Students
    0
      Name : "Amit Goenka"
      Major : "Physics"
    1
      Name : "Smita Pallod"
      Major : "Chemistry"
    2
      Name : "Rajeev Sen"
      Major : "Mathematics"
```

Name ▲	Value
Major	"Physics"
Name	"Amit Goenka"

Online validators :
http://jsonviewer.stack.hu/
https://jsonformatter.curiousconcept.com/
https://jsoneditoronline.org/

Retrieving data from a JSON files

Data in JSON files consist in a string of variable length (they can sometimes be enormous) which can be use with almost all programming languages.

Parsing the JSON data

In order to be able to access the data of such files, the data first needs to be parsed, which can be done using the function JSON.parse().

Retrieving data

Once an object contain the parsed JSON data has been created, it is possible to retrieve specific values associated with the different keys using the following syntax : object.key.

```
<h2> </h2>

<script>
let myJson = '{"name":"John", "age":30, "city":"New York"}';
let myObject = JSON.parse(myJson);

$("h2").html(myObject.name + ", " + myObject.age + ", " + myObject.city);
</script>
```

Explanation:
In the example above the variable myJson contains a JSON object mad of 3 key/value pairs (name, age and city). The parsed content of the JSON object is stored in the variable myObject. Finally, data are retrieved and displayed within the <h2></h2> using concatenation.

Retrieving data from external JSON files

The way data are retrieved from external JSON files is basically the same as we already have seen, but remember : external JSON files needs to be tested online or using a virtual server such as MAMP. Use the browser's inspector to make sure all is working fine. You may also have to specify the type in the opening script tag

Loading data from external JSON files

In order for the data to be available in the current document (this will seem a bit complicated at first but we'll see another option), a http request needs to be made to store the data in an object (a variable) using the function *XMLHttpRequest()*. The *open()* and *send()* function will also be used.

Retrieving data from an external JSON file using a http request

```
<h2></h2>

<script>
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                        var obj = JSON.parse(this.responseText);
                        $("h2").html(obj.prenom + " " + obj.nom + ", " + obj.age);
            }
};

xmlhttp.open("GET", "http://www.monamijean.com/cdi/mytest.json", true);
xmlhttp.send();
</script>
```

Retrieving data from an external JSON file using getJSON() function

```
<h1></h1>

<script type="text/javascript" language="javascript">
$.getJSON('mytest.json', function (data) {
            $("h1").append("You are " + data.age + " year old.");
});
</script>
```

Convert XML to JSON

One very easy way to convert XML data to JSON is to use a converter. Many are available online such as https://www.convertjson.com/xml-to-json.htm.

So, if we take the XML file that was used to lear XML in preceding lessons, here what it looks like once converted :

```
{                                                 // Start of object
        "book": [                                 // Array of all the books starts
                {                                 // First element of array starts
                "title": "Harry Potter",
                "author": "J K. Rowling",
                "year": "2005",
                "price": "29.99",
                "photo": "photos/hpotter.jpg"
                },                                // First element of array ends
                {
                "title": "L'avalée des avalés",
                "author": "Réjean Ducharme",
                "year": "1966",
                "price": "39.95",
                "photo": "photos/avale.jpg"
                },
                {
                "title": "Le petit prince",
                "author": "Antoine de Saint-Exupéry",
                "year": "1943",
                "price": "19.99",
                "photo": "photos/pprince.jpg"
                },
                {
                "title": "Les fleurs du mal",
                "author": "Charles Baudelaire",
                "year": "1857",
                "price": "49.95",
                "photo": "photos/fleurs.jpg"
                }
        ]                                         // Array of all the books ends
}                                                 // End of object
```

Retrieving JSON data from and array

Based on the preceding page's JSON file, see below how it is possible to extract all the data from an object's array in order to list them on screen.

```
<script type = "text/javascript" language="javascript">
$(document).ready(function() {

        $.getJSON('mytest4.json', function(data) {

        let x = data.book;
        let str = "";

        for (i=0; i < x.length; i++) {
                str += "<b>" + data.book[i].title + "</b>, " + data.book[i].author + ", " + data.
book[i].year + ", " + data.book[i].price + "<br />";

                $("div").html(str);
                }
        });

});
</script>
```

Explanation :

In a function created within *.getJSON* to store the content from an external file in =to a variable named *data*, the variable x is defined in order to use .length to evaluate how many children it contains. the variable str is created to store the final output. Finally, a loop runs for as long as there are children to output every element on a single line