Scripting languages II (LS2)

Class 10

JSON

Standing for *JavaScript Object Notation*, JSON is a human readable format for structuring data. It is primarily used to transmit data between a server and web application. It is an alternative to the very well known XML. Testing these files often require them to be hosted or to use a virtual server such as MAMP.

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.

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"
```

Syntax and structure of JSON files

In a JSON file, braces are used to define an object which can contain many key/value pairs.

{...}
Brace brackets are used to define an object.

[...]
Brackets are used to define an array.

N.B.: Commas are used to separate elements of an object
(no comma after the last element or the element will not be valid).

```
{
"name" : "John",
"age" : 30,
"city" : ["Montreal", "New York", "Paris"]
}

OR

{"name" : "John", "age" : 30, "city" : "New York"}
```

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*.

Retrieving specific data from JSON

```
<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.

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*.

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

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

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>
```

jQuery function each()

This function is used to loop through each element of a targeted object. You will find this function very useful for multi-element DOM manipulation, looping arbitrary arrays, and object properties.

Retrieving an element from an array

As you may most probably remember, it is quite easy to retrieve an element from an array. All there is to do is to specify its index number after the object's name.

```
<h2></h2>

<script>
let myArray = ["Dog", "Cat", "Bird", "Fish"];

$("h2").append(myArray[1]);                    // Would show "Cat"
</script>
```

Retrieving all the elements from an array

Of course, it would be possible to use a JavaScript loop (such as for) and the *length* method to show all the array's elements. Although, the function each() retrieves all the element contained in the array faster and in a much more simple way than an ordinary loop, as shown below.

```
<h2></h2>

<script>
let myArray = ["Dog", "Cat", "Bird", "Fish"];

$("h2").each(function(){
        $("h2").append(myArray);
});
</script>
```

Retrieving all the elements from a JSON file

Here, in order to retrieve the data from a JSON file, it is necessary to combine the function allowing to retrieve JSON data from an external file to the each function in order to show both keys and values.

JSON file :

```
{
"name1":"John",
"name2":"Doe",
"age":"25"
}
```

HTML file :

```
<div> </div>

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

        $.getJSON('mytest3.json', function(data) {

                $.each(data, function () {
                        $.each(this, function (name, value) {
                                $('div').append(name + value);
                        });
                });
        });
});
</script>
```

Result :

```
0J1o2h3n0D1o2e0215
```

Explanation :

In the example above, the function *$.getJSON()* makes ti possible to import JSON data from an external file to store in in an object named *data*. The *each()* function creates a loop making it possible to show all key/value pairs using the objects named *name* and *value* in a custom function. The result would show the index numbers and the associated characters of each of the values : 0J1o2h3n 0D1o2e 0215.

Removing name (object associated to the index number) would result in showing only values data : JohnDoe25