Advanced Internet Programming

Project no. 6 – 20.03.30 – JSON Technology and its applications

Hubert Beczkowski, ID: 307290

Introduction

Given task is devoted to JSON, a technology used for making manipulation of the JavaScript objects easier. JSON stands for **J**ava**S**cript **O**bject **N**otation. It is a text format for storing and transporting data. JSON is self-descriptive and easy to understand.

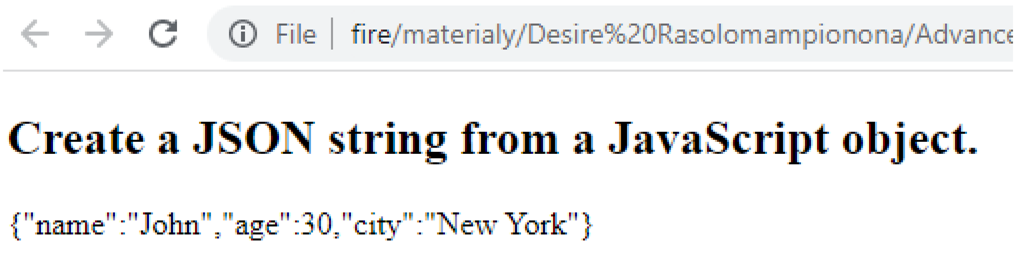
The JSON format is similar to code that creates JavaScript objects – thanks to that, a JavaScript program can easily convert JSON data into JavaScript Objects.

JavaScript has a built in function to convert JSON strings into JavaScript objects: JSON.parse(), while also it can convert an object into a JSON string: JSON.stringify().

Task

The task at hand is to analyze and explain the examples given in the course files. The assigned exercises were 3, 7 and 14.

- ***Example 3***

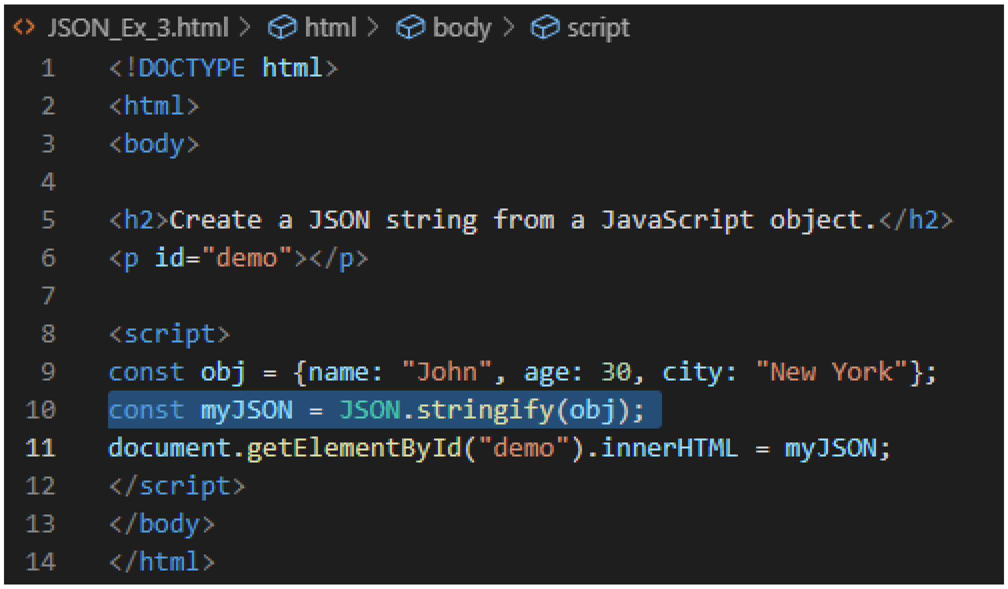

File | fire/materialy/Desire%20Rasolomampionona/Advanc

Create a JSON string from a JavaScript object.

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

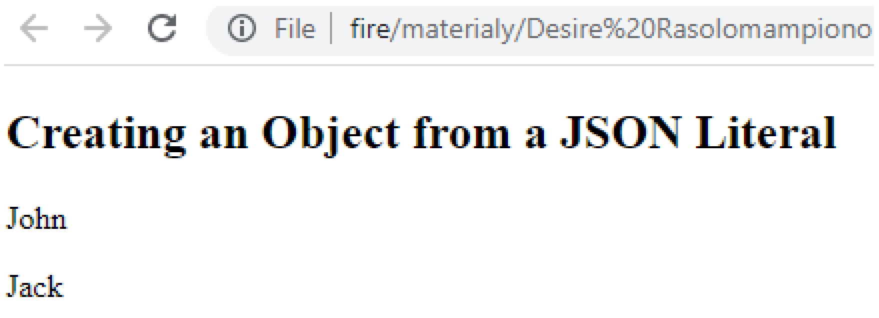
The html page of Example 3

In the first example, it is named as creating a JSON string from a JavaScript object. At first look at the html page, I thought that there is a mistake in code, because it prints out a string which is not what we want most of the time, but looking at the code one can see the idea behind this page.


```
JSON_Ex_3.html > html > body > script
1   <!DOCTYPE html>
2   <html>
3   <body>
4
5   <h2>Create a JSON string from a JavaScript object.</h2>
6   <p id="demo"></p>
7
8   <script>
9   const obj = {name: "John", age: 30, city: "New York"};
10  const myJSON = JSON.stringify(obj);
11  document.getElementById("demo").innerHTML = myJSON;
12  </script>
13  </body>
14  </html>
```
The code for Ex.3

As it is visible, the html code for exercise 3 consists of a constant object which creates fields “name”,”age” and “city”. The highlighted line 10 uses the JSON.stringify function, which converts given object (in our case, object is named “obj”) to a JSON string. The ability to convert a JavaScript object into a JSON string is useful in case one needs to send a JavaScript object across the network.

- ***Example 7***


File | fire/materialy/Desire%20Rasolomampiono

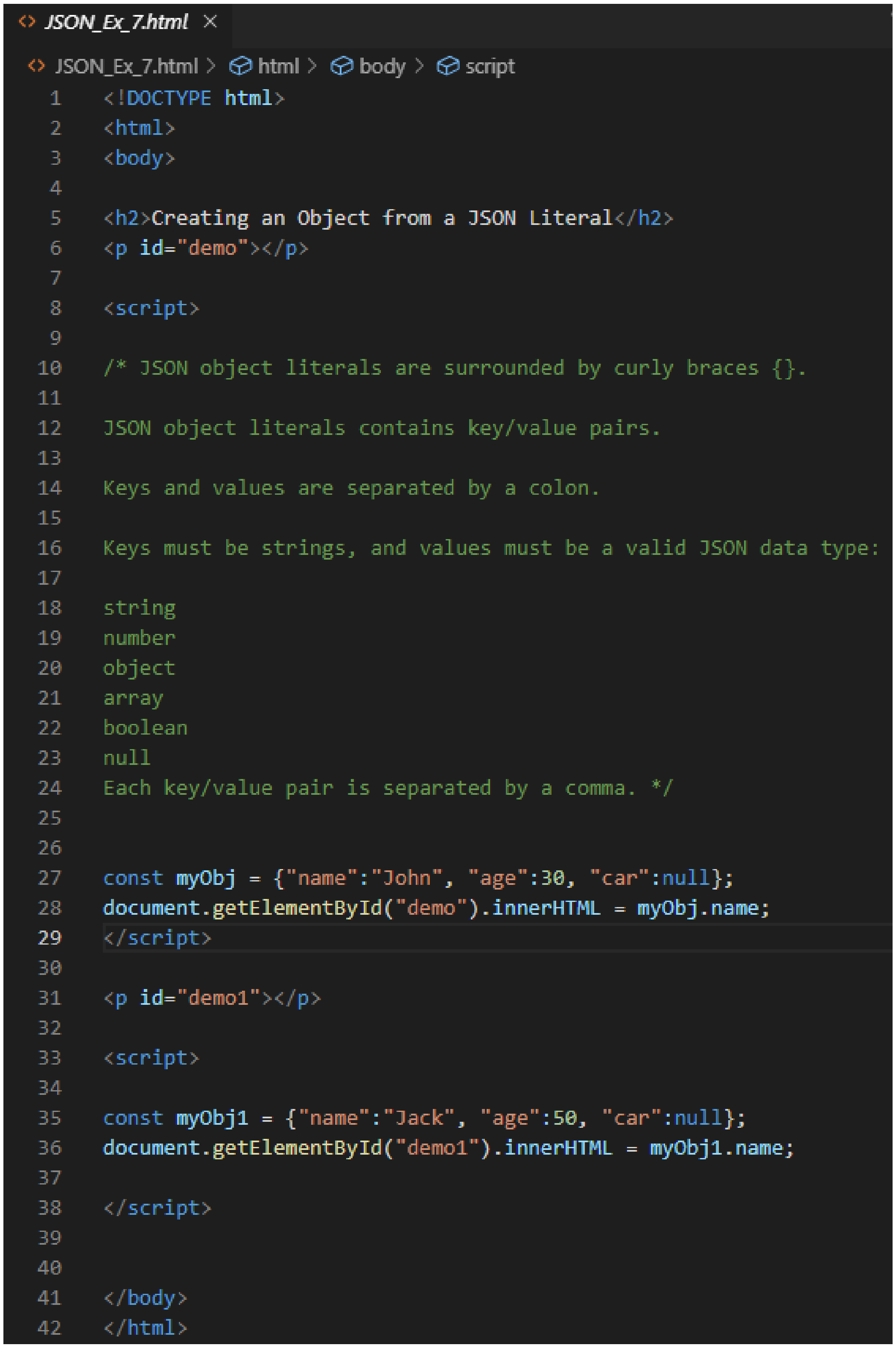
Creating an Object from a JSON Literal

John

Jack

The html page of ex.7

Looking at the html page it seems simple and not complicated – its just a page containing two names. Although, the code is much more interesting – it consists of objects (in this case, they are called “myObj” in the code) which are later taken by ID and printed on the page.



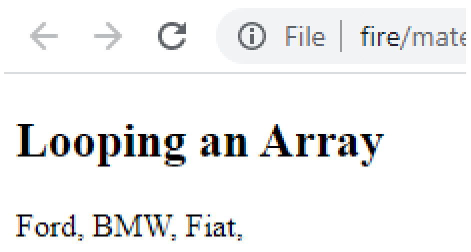
```
<!DOCTYPE html>
<html>
<body>

<h2>Creating an Object from a JSON Literal</h2>
<p id="demo"></p>

<script>

/* JSON object literals are surrounded by curly braces {}.

JSON object literals contains key/value pairs.

Keys and values are separated by a colon.

Keys must be strings, and values must be a valid JSON data type:

string
number
object
array
boolean
null
Each key/value pair is separated by a comma. */


const myObj = {"name":"John", "age":30, "car":null};
document.getElementById("demo").innerHTML = myObj.name;
</script>

<p id="demo1"></p>

<script>

const myObj1 = {"name":"Jack", "age":50, "car":null};
document.getElementById("demo1").innerHTML = myObj1.name;

</script>


</body>
</html>
```

The code of Ex.7

The objects in literals contain of two parts – keys and values. Introduction of such method allows to create a key which has assigned value, so that the key's value can be easily recalled, by getting the element by ID, and adding the key to the ID after the dot.

In given example, we are given two objects, which consist of the same keys: name, age and car, but different values. In a code written in that way, one can invoke the key values from the objects easily. In our case, the values were name ("myObj.**name**", "myObj1.**name**"), but if one were to assign it a different key (ex. "myObj.**age**", "myObj1.**car**"), the output value would change to the one which was added to the myObj.

- ***Example 14***

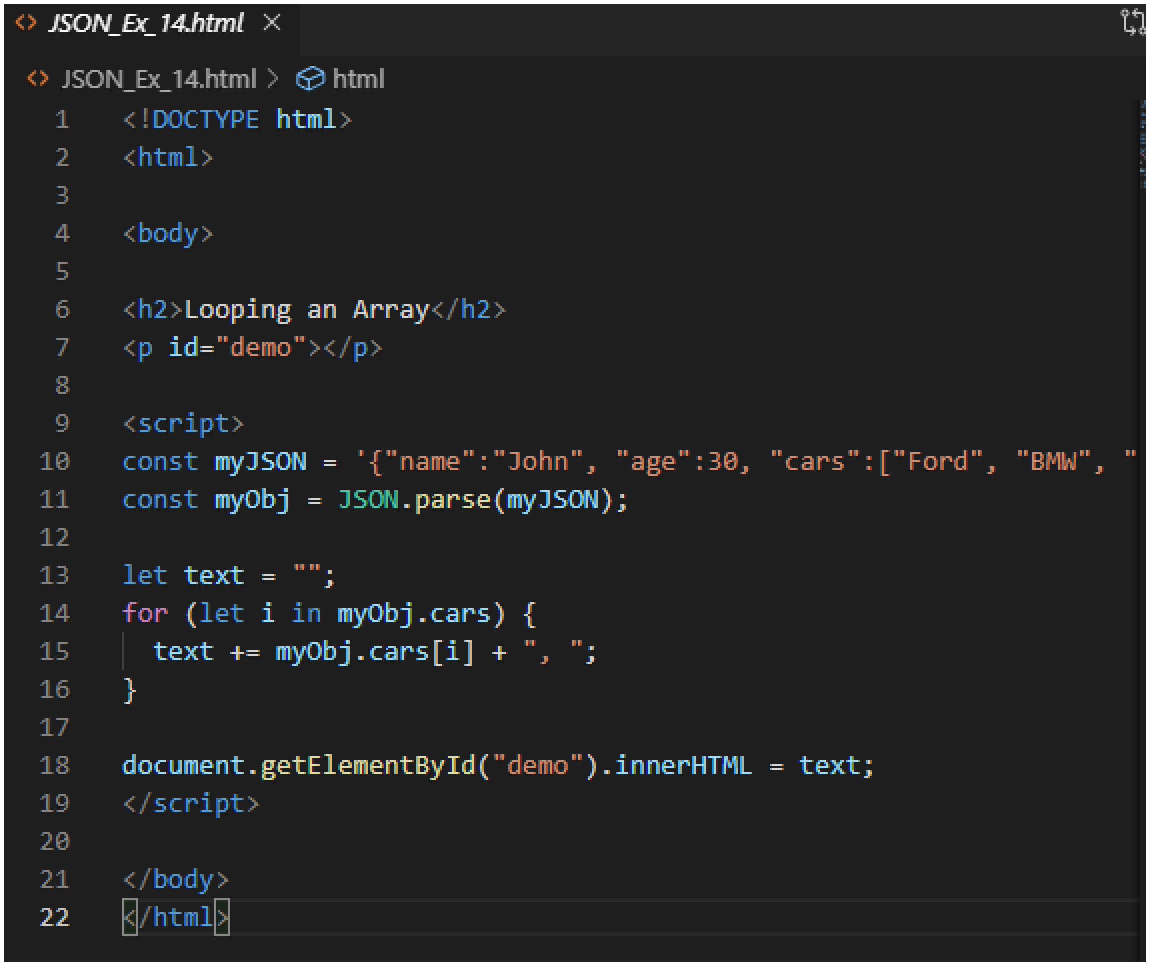


Looping an Array

Ford, BMW, Fiat,

The html page of ex.14

In example 14 we are introduced to looping an array with use of JSON technology. Looping through the array created in JSON allows one to iterate through each object in the array.



```
1   <!DOCTYPE html>
2   <html>
3
4   <body>
5
6   <h2>Looping an Array</h2>
7   <p id="demo"></p>
8
9   <script>
10  const myJSON = '{"name":"John", "age":30, "cars":["Ford", "BMW", "
11  const myObj = JSON.parse(myJSON);
12
13  let text = "";
14  for (let i in myObj.cars) {
15    text += myObj.cars[i] + ", ";
16  }
17
18  document.getElementById("demo").innerHTML = text;
19  </script>
20
21  </body>
22  </html>
```

The code of ex.14

In line 10, the myJSON is described, and we can see that the cars key has values in an array. It is further converted into JavaScript object using JSON.parse(myJSON) so that we can create for loop later.

In code, the text is created, for which a for loop is designed. The loop iterates through all the values in cars array, and adds them to the text. Later, the loop is called upon by the element ID and printed out on the html page.

Looping through an array makes calling values from it easier. We have to add less code, and obtain the same information. It automatizes the process of obtaining the data and printing it on the page.