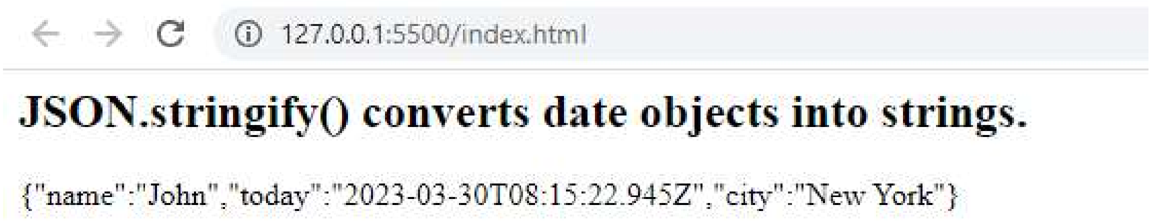
Report Lab n.6 (Task 5,7,12)

Task n.5

Code:

```
<html>
<body>



<h2>JSON.stringify() converts date objects into strings.</h2>
<p id="demo"></p>

<script>

/* In JSON, date objects are not allowed. The JSON.stringify() function will conv
ert any dates into strings. */

const obj = {name: "John", today: new Date(), city: "New York"};
const myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
</script>

</body>
</html>
```

Result:



JSON.stringify() converts date objects into strings.

```
{"name":"John","today":"2023-03-30T08:15:22.945Z","city":"New York"}
```

Description:

As we can see in this code is firstly declared an object (obj) with some attribute of which 2 with a string value and one with a new object as a value:

JavaScript Date objects represent a single moment in time in a platform-independent format. Date objects encapsulate an integral number that represents milliseconds since the midnight at the beginning of January 1, 1970, UTC.

The obj created is converted in string format by a JSON method : stringify().

And assigned to a new constant (myJSON).

At the end the myJSON will be displayed in the innerHTML of the paragraph with “demo” id.

As we can see from the result :

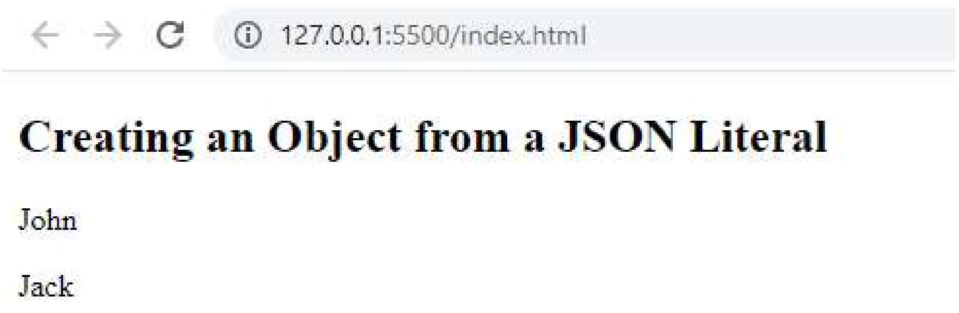
In JSON, date objects are not allowed. The JSON.stringify() function will convert an y dates into strings.

Task n.7

Code:

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

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

<script>

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>
```

Result:


127.0.0.1:5500/index.html

Creating an Object from a JSON Literal

John

Jack

Description:

As we can see the syntax used to declare the constant (myObj) is the JSON Object Literal syntax and it is also possible to create an object in JavaScript with that syntax in fact in the line after the declaration we see the usage of the constant myObj as an object, taking the “name” attribute (that in the first demo is “Jhon” and in the second one “Jack”) and displaying it respectively in the innerHTML of the paragraphs with “demo” and “demo1” as id.

From this example we can also see that:

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

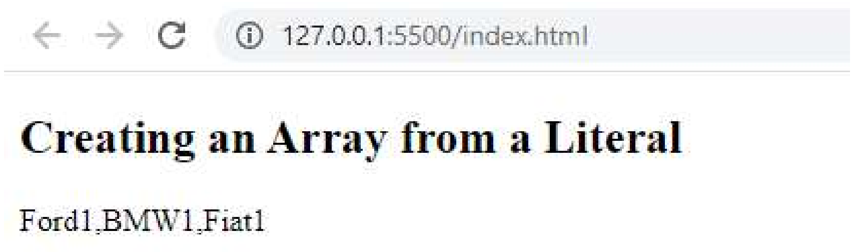
Each key/value pair is separated by a comma.

Task n.12

Code:

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

<body>

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

<script>


const myJSON = '["Ford1", "BMW1", "Fiat1"]';
const myArray = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myArray;

</script>

</body>
</html>
```

Result:



Creating an Array from a Literal

```
Ford1,BMW1,Fiat1
```

Description:

As we can see the first const declared is a Jason string

```
'["Ford1", "BMW1", "Fiat1"]'
```

With the line after it is assigned to myArray constant the result of JSON.parse() function that convert the myJSON JSON string into a JavaScript Object, but in this case:

When using the JSON.parse() on a JSON derived from an array, the method will return a JavaScript array, instead of a JavaScript object.

At the end the myArray will be displayed in the innerHTML of the paragraph with “demo” id.